Universitetet i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialization: Construction and Materials, specialization within Mechanical Construction	Spring semester, 2017 Open
Author: Svein Erik Nuland	Svein Erik Nuland …………………………………… (signature of author)
Supervisor(s): Assoc. Prof. Siri M. Kalvig Co-Supervisor: Knut Erik Teigen Giljarhus, UiS External Co-Supervisor: Jean-Baptiste Curien, Gwind	
Title of master's thesis: Wind Flow Simulation over Fish Farm Feed Barge	
Credits: 30 ETCs	
Keywords: OpenFOAM ALM HAWT/VAWT Offshore Fish farm CFD	Number of pages: 84 + supplemental material/other: 15 Stavanger, 15.06.2017 date/year

Preface

When I started to work on this thesis for exactly six months ago, I had no idea that I would learn so much in the coming months. After working hard so long, I can't imagine that it soon over. This has been a great experience from day one and I have a few people to thank for that.

Firstly, I would like to thank all my supervisors; Siri M. Kalvig for providing me with the opportunity to work with this project and as the leader of Research Network for Sustainable Energy for encouraging me to send an abstract for SOWE, and then to giving me the opportunity to travel to Spain to present my work. Knut Erik Teigen Giljarhus for providing me with good insight regarding CFD. Jean-Baptiste Curien for bringing me along to meetings, openness about the renewable energy business and for being a great support throughout this work, I really do appreciate it.

I would also thank; Ingrid Kristina Feyling for providing me with insightful comments and feedback, I do not know how my thesis would look like without it. Lorenzo Riva for the long and serious talks about the CFD model.

My partner Ida Maria has provided support and motivation not only these last moths, but for the last two years, I could not have done it without her. Lastly, I would like to thank all my family and friends for encouraging and supporting me along the way.

Abstract

There are approximate over 1000 fish farms in Norway, where half are connected to the grid and the rest are driven by diesel generators. Fish farms use large air compressors to feed the fish, which creates high power consumption. To reduce the diesel consumption and CO_2 emissions, created by the compressors, there are companies that specialize in providing green energy solutions. Gwind is a Stavanger based energy company that provide off-grid energy for this exact purpose. A master study done by H. Syse showed that a hybrid system with wind turbines, PV, Li-Ion batteries and two diesel generators over a 20-year period would reduce the CO_2 emissions and lower the diesel consumption.

Investigation of local wind flow and power generation with a wind turbine linked to the fish farm feed barge, was performed using the open source computational fluid dynamics (CFD) software OpenFOAM. The wind turbine is a vertical axis wind turbine (VAWT), modeled by an actuator line model (ALM). The ALM has been implemented with the use of a library called *turbinesFoam.*

A framework for wind flow simulations over fish farm feed barges has been developed. This framework includes a ALM of a VAWT, simulated with OpenFOAM's *pimpleFoam* solver, and *k-epsilon* turbulence model. The inlet is enriched with atmospheric boundary layer.

The framework has been used on two fish farm cases, Tallaksholmen and Nordheim. These are owned by Grieg Seafood Rogaland, and in collaboration with Gwind a wind measurement campaign was conducted, and cross-referenced with nearby wind stations to set an approximately real inflow condition.

The framework was used to investigate the optimal height placement of the VAWT on Tallaksholmen, coupled with a cost benefit analysis. To show the flexibility of the framework the second fish farm case, Nordheim, was setup and ready to run within a few hours. In this case the performance was increased, as a result of investigating the local wind flow before activating the turbine.

Based on the results of this study, it is recommended to install a VAWT on the Tallaksholmen fish farm feed barge. The operational performance should be compared against the simulations to further verify the computational approach.

List of symbols

Roman letters

A_{elem}	blade platform area	$[m^2]$
C_P	power coefficient	$[-]$
C_T	torque/trust coefficient	$[-]$
C_d	drag coefficient	$[-]$
C_l	lift coefficient	$[-]$
$C_\mu, C_{\epsilon 1}, C_{\epsilon 2}$	constants in $k - \epsilon\ model$	$[-]$
D_f, F_d	drag force	$[N]$
L_f, F_l	lift force	$[N]$
U^*	friction velocity	$[m/s]$
d^+, y^+	normal distance to the wall	$[-]$
k_{rf}	reduced frequency	$[-]$
z_0	surface roughness length	$[m]$
z_g	minimum z-coordinate	$[m]$
$\boldsymbol{f}_b$	body forces	$[-]$
h	height	$[m]$
A	projected area	$[m^2]$
B	property of fluid	$[-]$
Co	Courant number	$[-]$
L	characteristic length	$[m]$
M	pitching moment	$[Nm]$
R	largest radius	$[m]$
T	torque/trust	$[Nm]/[N]$
V	control volume	$[-]$
$V, U, \mathbf{v}$	velocity	$[m/s]$
a	axial flow induction factor	$[m]$
b	intensive value of the property of fluid	$[-]$
c	cord length	$[m]$

g	gravity vector	$[m/s^2]$
k	turbulent kinetic energy	$[m^2/s^2]$
p	pressure	$[N/s^2]$
r	radius	$[m]$
t	time	$[s]$
x, y, z	Cartesian coordinates	$[m]$
$\boldsymbol{I}$	identity tensor	$[-]$

Greek letters

C_{mesh}	e_{mesh} parameter	$[-]$
N_{elem}	number of elements	$[-]$
λ_b	bulk viscosity coefficient	$[-]$
μ_t	turbulent viscosity	$[kg/sm]$
$\sigma_\epsilon, \sigma_k$	constants in $k - \epsilon\ model$	$[-]$
τ_w	wall shear stress	$[kg/ms^2]$
ϵ_{mesh}	Gaussian width parameter	$[m]$
$\boldsymbol{\tau}^R$	Reynolds stress tensor	$[-]$
∇	gradient	$[-]$
Δt	time step length	$[s]$
$\Delta x, \Delta y, \Delta z$	grid spacings	$[m]$
v	kinematic viscosity	$[m^2/s]$
α	angle of attack	$[-]$
κ	von Karmann's constant	$[-]$
λ	tip speed ratio	$[-]$
μ	viscosity	$[kg/sm]$
ρ	Density	$[kg/m^3]$
ω	angular rotor velocity	$[rad/s]$
ϵ	rate of viscous dissipation of k	$[m^2/s^3]$
ϕ	flow variable	$[-]$

Superscript

$'$	fluctuating value
$\boldsymbol{T}$	transposed
$+$	dimensionless

Subscript

∞	upstream
D	disk
max	maximum value
rel	relative
W	far wake

Abbreviations

ALM	Actuator line model
CAD	Computer Aided design
CFD	Computational fluid dynamics
Cp	Power coefficient [%]
DFIG	Doubly Fed Induction Generator
FVM	Finite Volume Method
GAMG	Generalized method of geometric-algebraic multi-grid
GSFR	Grieg Seafood Rogaland
HAWT	Horizontal axial wind turbine
HHTSL	Horizontally Homogenous Turbulent Surface Layer
O&M	Operation and Maintenance
OpenFOAM	Open Source Field Operation and Manipulation
PISO	Pressure Implicit with Splitting of Operators
PV	Photovoltaics
RANS	Reynolds averaged Navier-Stokes

SCIG	Squirrel Cage Induction Generator
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
STL	stereolithography file
TKE	Turbulent kinetic energy
TSR	Tip speed ratio
VAWT	Vertical axis wind turbine

Contents

1 Introduction

There are approximately over 1000 fish farms in Norway [1], where half are connected to the grid and the rest are driven by diesel generators. Fish farms use large air compressors to feed the fish, which creates high power consumption. To reduce the diesel consumption and CO_2 emissions, created by the compressors, there are companies that specialize in providing green energy solutions. Gwind is a Stavanger based energy company that provide off-grid energy for this exact purpose [2]. A master study done by Syse showed that a hybrid system with wind turbines, photovoltaics (PV), lithium-ion batteries and two diesel generators over a 20-year period would reduce the CO_2 emissions and lower the diesel consumption [3].

To investigate wind flow over the feed barge and power generation with a wind turbine linked to the feed barge, flow simulations are performed using the open source computational fluid dynamics (CFD) software OpenFOAM. The wind turbine is a vertical axis wind turbine (VAWT), modeled by an actuator line model (ALM). The ALM has been implemented with the use of a library called *turbinesFoam* [4]. This study uses Reynolds averaged Navier-Stokes (RANS) for the governing equations, and the turbulence is modelled with the $k - \epsilon$ turbulence model.

1.1 Objectives

The main objectives of this thesis are to construct a framework for wind flow simulations for fish farm feed barges, and to inspect local wind over fish farms to find optimal placement of the turbine. Additionally, the optimal height correlated with cost benefit will be considered. This work will hopefully act as a stepping stone to further investigating wind flow simulations over fish farm feed barges, considering floating turbines and implementation of multiple turbines to construct a small turbine park.

1.2 Thesis layout

This thesis starts with the introduction of offshore fish farming, and highlight the energy consumption during the winter and summer season. The feeding process is also introduced.

In chapter 3, relevant theory will be introduced. First, theory regarding wind energy will be introduced. The actuator disk theory will be used to explain the basics physics of a wind turbine i.e. how much power that can be extracted from the wind. Next, an introduction to the fundamentals of computational fluid dynamics (CFD) is given. The governing equations and turbulence modeling will be introduced. The open source CFD library OpenFOAM is presented. Lastly, the ALM that has been used will be introduced. The theory and source code will be reviewed, and the outputs of the model will be discussed.

Next is chapter 0, Gwind and Grieg Seafood Rogaland is introduced. A wind measurement campaign that was conducted in January on Tallaksholmen, will be presented. In chapter 5, modelling approach and experimental setup is introduced. This chapter includes geometric modelling, mesh generation and boundary conditions.

In chapter 6, results are presented and discussed when presented. Lastly, in chapter 7 the conclusion will be presented and the further works will be discussed.

1.3 Prior works

This thesis is a further inspection of what Syse concluded with in investigating off-grid energy solutions for the salmon farming industry [3]. He concluded that a hybrid energy system for offshore salmon farms would be feasible, this would lower the cost and reduced the CO_2 emissions. A hybrid system containing the capacity of; $14\,kW$ wind turbine, $35\,kW$ PV, $146\,kWh$ lithium-ion batteries and two diesel generators, $130\,kW$ and $10\,kW$, would be used to meet the average energy consumption of $341.92\,kWh$ per day. This thesis will look at potential power output from VAWT, with the use of CFD.

The VAWT turbine is being modeled with the use of an ALM developed by Bachant, Goude and Wosnik [5]. They developed a OpenFOAM library called *turbinesFoam* [4]. The ALM was made to bridge the gap between high and low fidelity numerical modeling tool for VAWT. The model has been implemented in this thesis work, due to the low fidelity numerical modeling it is possible to implement in a flow simulation over a fish farm feed barge, without too much computational cost.

The ALM is simultaneously being used in another master study by Riva [6]. His study aims to study the feasible use of the ALM for build-environment VAWT's. The build-environment he is studying is the University of Stavanger campus, where he implemented the VAWT on the building roof. Riva considered optimization of, wind velocity, rotational speed and location of the turbine. Riva also investigated time and grid sensitivity analysis.

2 Offshore Fish Farming

Offshore fish farming started in the early 1970, the pioneers started with rainbow trout and salmon in simple net cages in the sea [7]. The industry have grown exponentially and is still growing, only in 2015 there was exported 1.3 million ton fresh and frozen salmon combined [8]. Aquaculture in Norway is a combination of salmon, rainbow trout, cod and shellfish as well as some other marine species [9]. Salmon farming is clearly the biggest share of the Norwegian aquaculture, numbers from 2015 states 94.5% [8].

Figure 1."En flott dag på Tinnlandet", picture of net cages from Grieg Seafood Finnmark [10]

This is an industry that faces a lot of challenges and have yet to overcome a lot of technical and socio-economic challenges to become (fully) renewable. Environmental problems such as, escaped salmon, lice and pollution is only a handful of the problems that exists in this industry [11], but with the technical advancements on the marine sector in recent years, the challenges are (diligently) being dealt with and both innovative and sustainable solutions are already available on the marked where both membrane and laser technology [12, 13].

There is a high demand to make fish farming more environmental friendly and one of the solutions is introduction of renewable energy. The introduction of wind turbines is one of the solutions that Syse presented in his studies [3]. This thesis looks further into the potential power output of horizontal- and vertical axis wind turbines.

Syse study showed how much energy that is being consumed during the feeding process, the next section there will be given a small review of the energy consumption of a fish farm feed barge based on Syse's studies.

2.1 Energy consumption

Energy consumption of a fish farm can be categorized into two; one during summer and one during winter. The consumption during the winter months will be higher because cage light used to reduce the plasma melatonin on the salmon [14].

The energy consumed on the barge during the night is mostly heating, lights, computers, surveillance system and the normal use of kitchen appliances.

During the day, the consumption is higher due to the feeding process, which will be discussed in the next chapter. Both crane and the dead fish handling system is utilized during the day. A typical daily energy distribution profile is shown in Figure 2, for both summer and winter. The figure clearly shows the significant energy peaks during mid-day. Additionally, the winter months show an overall higher level of energy consumption, mainly due the cage lights mentioned above and heating of the barge.

The energy will generally have the form as shown in Figure 2:

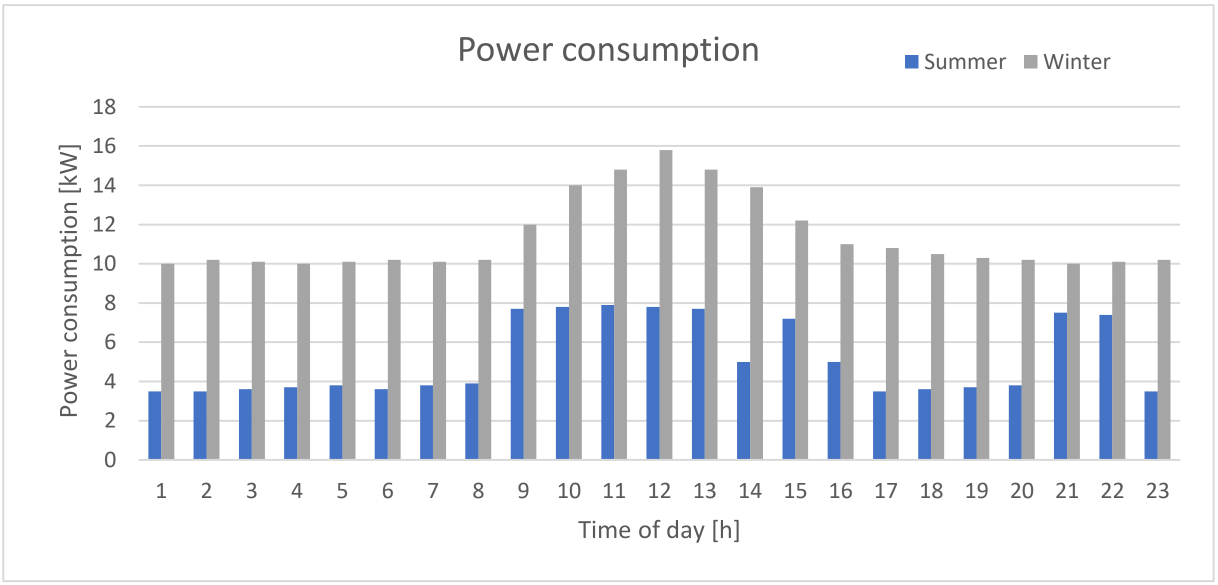


Figure 2. Power consumption of a fish farm barge during summer and winter

2.2 Feeding process

The salmon is fed according to the size of the fish. Table 1 show a typical feeding pattern throughout the growth cycle of a fish.

Table 1. Typical feeding patterns throughout the growth cycle [15]

Growth intervals	**0.1-0.2 kg**	**0.2-1 kg**	**1-2 kg**	**2-3 kg**	**3-4 kg**	**4-5 kg**
Feed consumption (Norway)	0.08 kg	0.75 kg	1.00 kg	1.05 kg	1.10 kg	1.20 kg
Time, months	2	4	4	3	2	2

A salmon requires approximate 0.87 kg in average of food on a daily basis over its lifetime.

Considering for instance a fish farm containing 1 million fish, the total amount of food per day approximates to 0.87 tons of food per day, when excluding losses. The food is stored in silos on the feed barge, where the capacity and number of silos depend on the size of the fish farm.

There are different solutions to transport the food pellets to the transport system. AKVA Group have solutions that use feed dosers valves that can be connected in series [16]. [17]

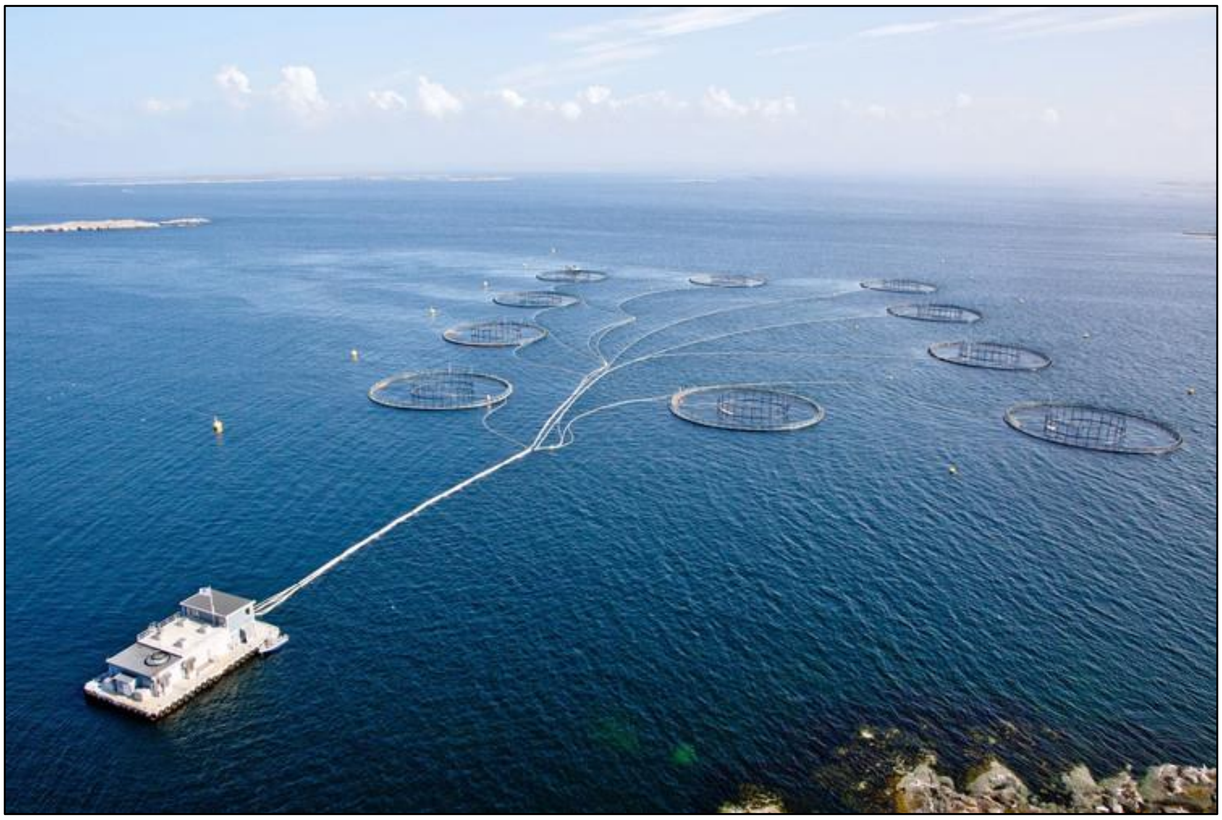


Figure 3. A barge monitors and pumps feed to salmon in net pens at Marine Harvest farm in Norway [17]

According to Syse’s report, GSFR uses progressive cavity pumps. The pellets are then blown out, with optimal speed to prevent blocking or too fast feeding, to the net cages using compressed air. The transport pipes can vary in diameter and length see Figure 3. The pellets are then spread uniformly into the net cage using a surface spreader or a subsea feeder.

In continuation of Syse’s work and the challenges he addressed, this thesis will investigate the potential power output from VAWT, with the use of CFD.

3 Theory

3.1 Wind Energy

In this chapter, theory regarding wind energy is introduced. The actuator disc theory has been used to explain the basics physics of a wind turbine. Two-dimensional aerodynamics will be presented. A wind turbine is defined as a machine which converts power in the wind into electricity [18].

The basic working principle of a wind turbine is that the aerodynamic force of lift produces a net positive torque on a rotating shaft. This produces first mechanical power and then the power is transformed to electricity through a generator.

Wind is not something that one can store, so the wind turbine creates have an output that fluctuate with the actual wind at the turbine.

3.1.1 Structure of wind turbine

To understand the basics of how a wind turbine operates, the main components of a horizontal axis wind turbine (HAWT) are mentioned with their function.

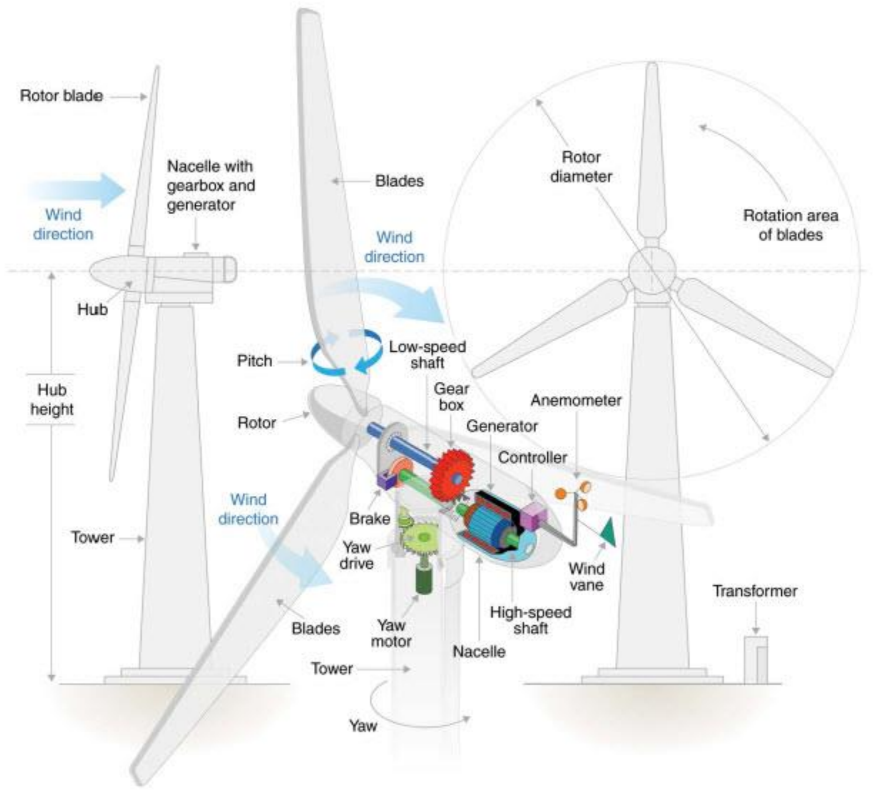


Figure 4. Components of a modern HAWT with gearbox. [19]

Rotor

The rotor is a combination of the hub and the blades [20]. These components are very important when the performance and overall cost is considered. Pitch control, most used in larger turbines and some small turbines, are used maximize power generated while minimizing loads [21].

Drive train

The drive train typically includes the gearbox, high- and low-speed shaft, support bearings, couplings, a brake and the rotating part of the generator. The gearbox main purpose is to speed up the rotation of the low-speed shaft to much higher rounds per minute (rpm), to the high-speed shaft, suitable for driving a generator.

There are two types of gearboxes used in wind turbines, parallel shaft and planetary. Planetary gearboxes have advantages when the wind turbine gets larger, due to weight and size. There is also an increase in larger wind turbines which also can use multiple-generator drivetrains [22].

Since wind energy is the fastest growing renewable energy source, there is a need to reduce downtime and operation and maintenance (O&M) cost and increase the reliability to make the wind turbine more competitive. There have been done research on drivetrain condition monitoring and on increasing the reliability of the gearbox [23, 24].

Generator

The generator is selected based on if the wind turbine is to be used at variable speed, or directly connected to the grid. If the generator is connected to power electronic converters the turbine can operate at variable speed, an option is the doubly fed induction generator (DFIG). When the output of a wind turbine is more than 1MW a DFIG is necessary [25]. Where the wind turbine is directly connected to the grid a squirrel cage induction generator (SCIG) are often used. The SCIG have more narrow speed range but is more robust and not as costly as the DFIG.

Nacelle and yaw system

The nacelle is the housing of the wind turbine, which includes the machines main frame, and yaw orientation system. The mainframe provides mounting and proper alignment of the drive train components. The cover is for protecting the nacelle content [26].

The yaw orientation system is there to align the wind turbine to the wind. On upwind wind turbines, an active yaw drive is mostly used, this mechanism is controlled by a control system. One other alternative is the passive system, which self-align with the wind. The passive system requires a tail when used by an upwind wind turbine, this is not required with a downwind wind turbine.

Tower and foundation

Tower and foundation have many different setups, but the more principal types are free-standing steel tubes, truss towers and concrete towers. Smaller turbines also use guyed towers. The characteristics of the placements of the turbine greatly influence the tower and foundation selection [18]. One of the more important things is the stiffness of the tower, dynamics responses of the tower should be calculated to avoid resonance. For offshore floating wind turbines the global dynamics are even more important [27].

Controls

The control system for a wind turbine is the most vital component considering power production and machine operation [18]. The control system is an assembly of components, such as; **Sensors** for speed, position, flow, temperature current, voltage etc. **Controllers** for mechanical mechanisms and electrical circuits. **Power amplifiers** for switches, electrical amplifiers, hydraulic pumps and valves. **Actuators** for motors, pistons, magnets, and solenoids. **Intelligence** components such as computers and microprocessors.

The control system is specific for each wind turbine, the main objectives for the control system are, setting limits for the torque and power experienced by the drive train. Maximizing the fatigue life for the different components and also maximizing the energy production [18].

3.1.2 2D Aerodynamics

The wind turbine blade is often long and narrow and the spanwise velocity component is much lower than the streamwise component, therefore the aerodynamic models that the flow at a given radial position is two dimensional and the 2D airfoil data can be used [28]. To generalize, an airfoil for an aircraft is designed to generate the most lift and as little drag as possible, making the lift-drag ratio as high as possible. The definition of lift and drag can be seen in Figure 5a.

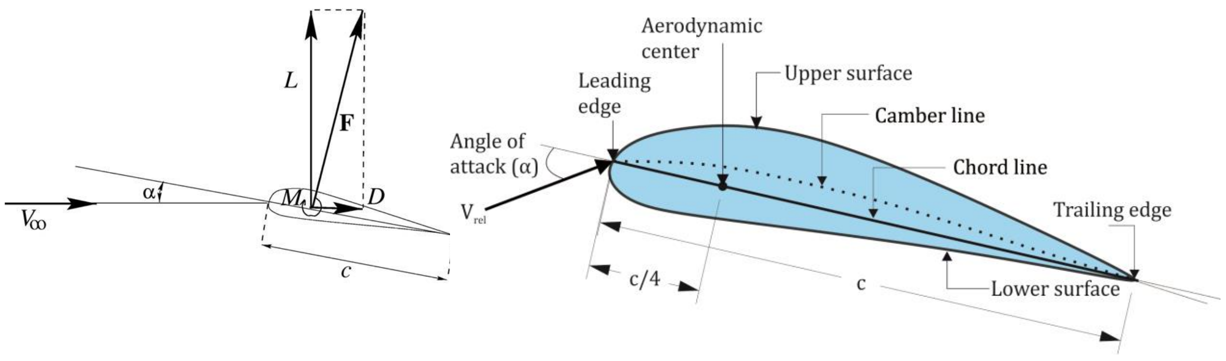


Figure 5. Definition of lift, drag (a) and airfoil nomenclature (b); Vertical force, L, are the lift force. Horizontal forces, V and D, are the relative velocity and drag force respectively. α are the angle of attack. c is the cord length. The moment, M, in this case located at the quarter cord. The reacting force from the flow are F. [28, 29]

Lift and drag coefficients C_l and C_d are defined as:

$$C_l = \frac{L_f}{\frac{1}{2} * \rho * V_{rel}^2 * c} = \frac{Lift\ force}{Dynamic\ force} \qquad \text{Equation 3.1}$$

$$C_d = \frac{D_f}{\frac{1}{2} * \rho * V_{rel}^2 * c} = \frac{Drag\ force}{Dynamic\ force} \qquad \text{Equation 3.2}$$

where ρ is the density and c is the length of the airfoil. The cord line is defined as the line from the leading edge. To describe the forces completely it is necessary to know the moment M, as shown in Figure 5a, about a point on the airfoil. This point is often at located at the quarter cord, also called aerodynamic center, which is one fourth length on the cord line from the leading edge. The pitching moment coefficient C_m is defined as:

$$C_m = \frac{M}{\frac{1}{2} * \rho * V_{rel}^2 * A * c} = \frac{Pitching\ moment}{Dynamic\ moment} \qquad \text{Equation 3.3}$$

where A is the projected airfoil area ($cord * span$).

When the moment is positive as seen in Figure 5a and then acts in the aerodynamic center the leading edge turns upwards [28].

If one looks at the physical explanation of the lift is that the shape of the airfoil forces streamlines to curve around the geometry, when it curves its velocity increases and pressure decreases see Equation 3.4, governed from Euler equation for inviscid fluid flow:

$$\frac{\partial p}{\partial r} = \frac{\rho V^2}{r}$$ Equation 3.4

Where $\partial p/\partial r$ is the pressure gradient, r the curvature of the streamline and V the velocity.

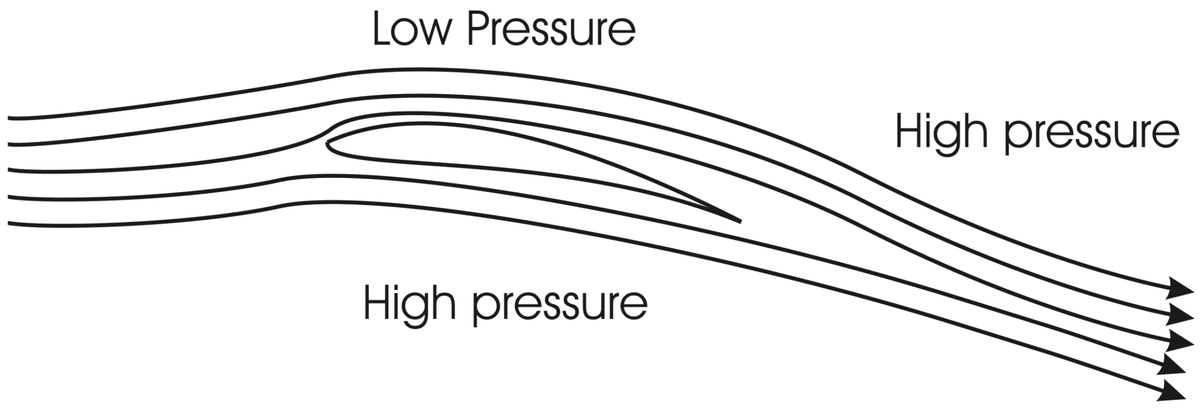


Figure 6. Streamlines over an airfoil [30]

The streamlines under the airfoil are less curved than the streamlines that goes above the airfoil, so the velocity is not that much changed which results in a high-pressure gradient below, and the opposite above the airfoil. This results in a positive lift [30].

The effect of Reynolds number and angle of attack

The Reynolds number (Re) has a significant importance on the behavior of the foils. The Reynolds number, Re, is defined by:

$$Re = \frac{UL}{\nu} = \frac{\rho UL}{\mu} = \frac{Inertial\ force}{Viscous\ force}$$ Equation 3.5

where ρ is the fluid density, μ is fluid viscosity, $\nu = \mu/\rho$ is the kinematic viscosity, and U and L are velocity and the length that characterize the scale of the flow. Equation 3.5 states that when the Re decrease the relative magnitude of the viscous forces become more than the inertial forces. Then the surface friction and pressure gradients increases. The result of this is that the drag coefficient increases and the lift is reduced [18].

In symmetric foils, like the NACA 0021, when the angle of attack (α) is parallel to the cord ($\alpha = 0°$) the lift coefficient is zero. At low angle of attack, the lift coefficient is small and increases linearly with increasing the angle of attack. If a cambered foil is used the lift is increased at low α [31].

When α reaches a specific point, the performance of the airfoil drops and it stalls. At this point, one also has the maximum lift. The lift behavior of the foil is the same for negative angle of attacks [18, 31].

3.1.3 The Actuator disc theory

One of the more important parts of wind turbines is how much power one can extract from the wind and turn into electrical power, also the trust of the wind acting on the rotor and the effect the rotor operation has on the local wind field. In Figure 7 one can see how the energy is extracting streamtube of a wind turbine.

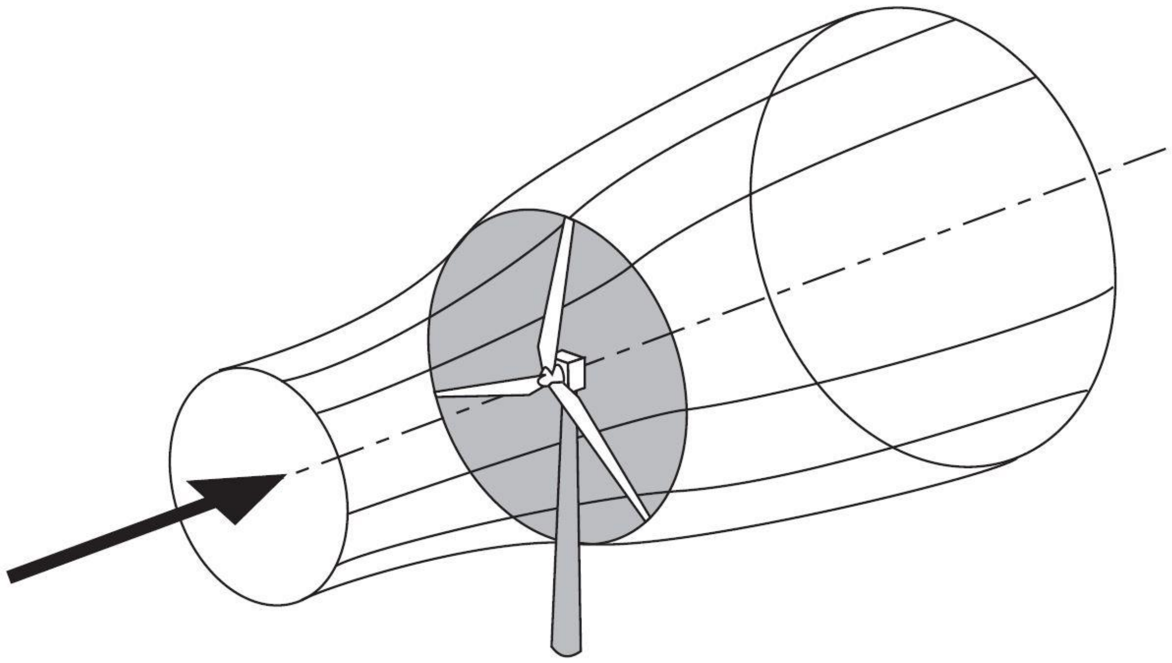


Figure 7. The energy extracting streamtube of a wind turbine [32]

Betz developed the global theory of wind machines at Göttingen institute on Germany in 1982, based on a linear momentum theory developed over 100 years ago that were once used to predict the performance of ship propellers. The fundamental equation was first introduced by Betz together with Lanchester, a British aeronautic pioneer in 1919 [33]. A short walkthrough of how Betz approached the analysis is conducted in this subchapter. The equations and formulations are inspired by [18, 32, 33].

The wind rotor is assumed to be an ideal energy converter, meaning that:

- It does not possess a hub
- It possesses an infinite number of rotor blades which do not result in any drag resistance to the wind flowing through them, no frictional drag

The analysis uses also these following assumptions:

- Homogenous, incompressible, steady state fluid flow
- Uniform thrust over the disc or rotor area
- A non-rotating wake
- The static pressure far upstream and far downstream of the rotor is equal to the undisturbed ambient static pressure.

From Figure 7 one can see an example on how the removal of kinetic energy from the wind affects the mass of air which passes through the rotor disc. Assuming that the affected mass of the air remains separate from the air which does not pass through the rotor disc and does not slow down, a boundary surface can be drawn containing the affected air mass and its boundary can be extended upstream as well as downstream forming a long streamtube of circular cross-section [32]. The air within the streamtube slows down, but is not compressed, the cross-sectional area of the stream tube must expand to accommodate the slower moving air as seen in Figure 7.

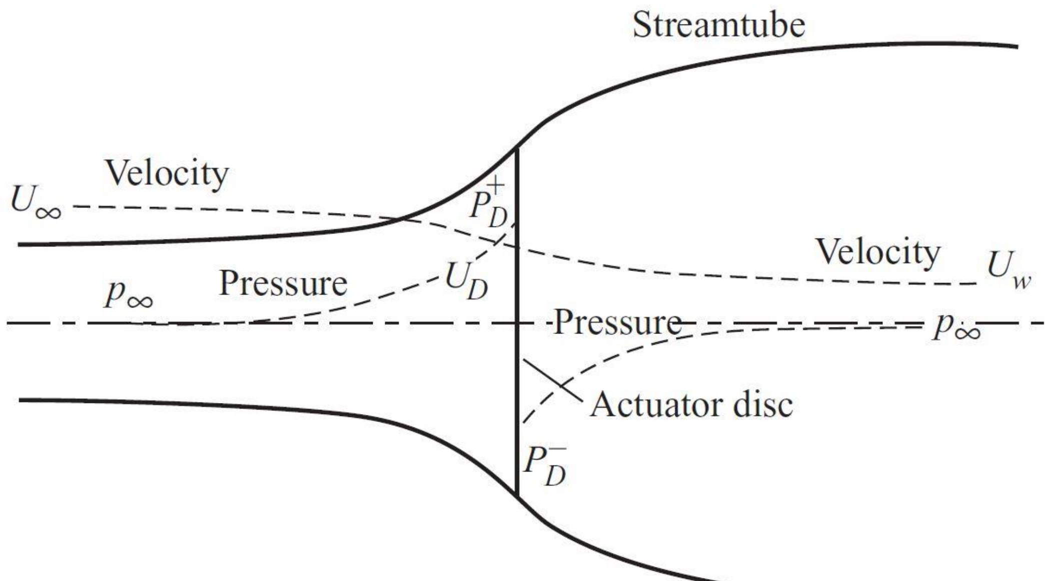


Figure 8. An energy extracting actuator disc and streamtube [32]

To describe how the energy is extracted inside the streamtube one can use the actuator disc shown in Figure 8.

The expansion of the streamtube is because of the mass flow rate must be the same upstream and downstream. The mass of air which passes through a given cross-section of the streamtube in a unit length of time is ρAU, where ρ is the air density, A is the cross-sectional area and U is the flow velocity. The mass flow rate must be the across the streamtube so that [32]

$$\rho A_\infty U_\infty = \rho A_D U_D = \rho A_W U_W \qquad \text{Equation 3.6}$$

where the symbol ∞ refers to condition far upstream, D refers to conditions at the disc and W refers to conditions in the far wake.

The stream-wise component of this induced flow at the disc is given by $-aU_\infty$, where a is called the axial flow induction factor, or the inflow factor. At the disc, therefore, the net stream-wise velocity is

$$U_D = U_\infty(1-a) \qquad \text{Equation 3.7}$$

Momentum theory

The air that passes through the disc undergoes an overall change in velocity, $U_\infty - U_W$ and a rate of change of momentum equal to the overall change of velocity times the mass flow rate:

$$Rate\ of\ change\ of\ momentum = (U_\infty - U_W)\rho A_D U_D \qquad \text{Equation 3.8}$$

The change of momentum comes entirely from pressure difference across the actuator disc, because the streamtube is surrounded by air which results in zero net force. Therefore,

$$(p_D^+ - p_D^-)A_D = (U_\infty - U_W)\rho A_D U_\infty(1-a) \qquad \text{Equation 3.9}$$

To obtain the pressure difference at the disc, Bernoulli's equation is applied separately to the upstream and downstream sections of the tube. Bernoulli's equation states that, under steady conditions, the total energy in the flow, compromising kinetic energy, static pressure energy and gravitational potential energy, remains constant provided no work is done on or by the fluid. For a unit volume of air,

$$\frac{1}{2}\rho U^2 + p + \rho g h = const$$ Equation 3.10

Upstream we have,

$$\frac{1}{2}\rho_\infty U_\infty^2 + p_\infty + \rho_\infty g h_\infty = \frac{1}{2}\rho_D U_D^2 + p_D^+ + \rho_D g h_D$$ Equation 3.11

Assuming incompressible flow, $(\rho_\infty = \rho_D)$ and horizontal $(h_\infty = h_D)$ then,

$$\frac{1}{2}\rho U_\infty^2 + p_\infty = \frac{1}{2}\rho U_D^2 + p_d^+$$ Equation 3.12

Similarly, downstream,

$$\frac{1}{2}\rho U_W^2 + p_\infty = \frac{1}{2}\rho U_D^2 + p_d^-$$ Equation 3.13

Subtracting Equation 3.11 and Equation 3.12,

$$(p_D^+ - p_D^-) = \frac{1}{2}\rho(U_\infty^2 - U_W^2)$$ Equation 3.14

Inserting Equation 3.14 into Equation 3.9 gives,

$$\frac{1}{2}\rho(U_\infty^2 - U_W^2)A_D = (U_\infty - U_W)\rho A_D U_\infty(1-a)$$ Equation 3.15

and so,

$$U_W = (1-2a)U_\infty$$ Equation 3.16

That is, half the axial speed loss in the streamtube takes place upstream of the actuator disc and half downstream.

Power coefficient

The force on the air becomes, from Equation 3.9

$$T = (p_D^+ - p_D^-)A_D = 2\rho A_D U_\infty^2(1-a)$$ Equation 3.17

This force is concentrated at the actuator disc, the rate of work done by the force is TU_D and hence the power extraction from the air is given by

$$Power = TU_D = 2\rho A_D U_\infty^3 a(1-a)^2 \qquad \text{Equation 3.18}$$

The power coefficient (C_P) is then defined as

$$C_P = \frac{Power}{\left(\frac{1}{2}\right)\rho U_\infty^3 A_D} \qquad \text{Equation 3.19}$$

where the denominator represents the power available in the air, in the absence of the actuator disc.

Therefore,

$$C_P = 4a(1-a)^2 \qquad \text{Equation 3.20}$$

The Lanchester-Betz limit

The Lanchester-Betz limit is the maximum C_P, to get the most out of the wind as possible one must derivative C_P, Equation 3.20, with respect to a and setting it equal to zero,

$$\frac{dC_P}{da} = 4(1-a)(1-3a) = 0 \qquad \text{Equation 3.21}$$

That gives a value of $a = 1/3$, therefore,

$$C_{P_{max}} = \frac{16}{27} = 0.593 \qquad \text{Equation 3.22}$$

This limit has not been exceeded yet. Given the basic laws of physics, this is the maximum power possible.

The thrust coefficient

The axial thrust on the disc is caused by the pressure drop, from Equation 3.17,

$$T = \frac{1}{2}\rho A_D U_\infty^2[4a(1-a)] \qquad \text{Equation 3.23}$$

This can also be characterized by a non-dimensional trust coefficient,

$$C_T = \frac{T}{\frac{1}{2}\rho U_\infty^2 A_D} = \frac{Thrust\ force}{Dynamic\ force} = 4a(1-a) \qquad \text{Equation 3.24}$$

The variation of power- and trust coefficient with the axial induction factor a is shown in Figure 9.

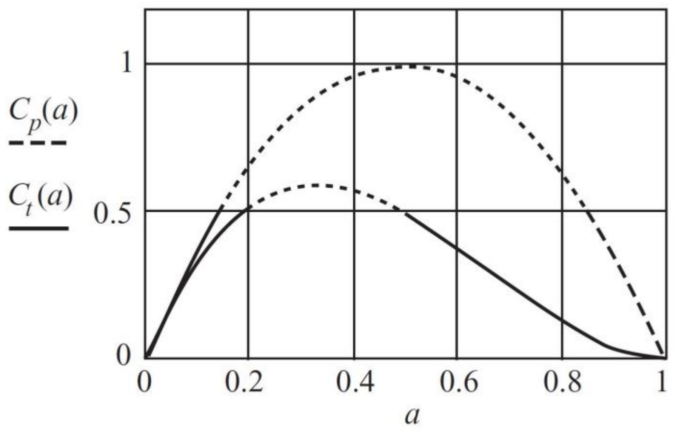


Figure 9. Variation of C_P and C_T with axial induction factor a [32]

The problem arises, when as one sees from Equation 3.24, that the C_T has a maximum of 1.0 when $a = 0.5$ the U_∞ becomes zero or negative. In these conditions, the momentum theory no longer applies and an empirical modification must be made, which will not be discussed in this paper [18, 32]. In practice when the axial induction factor approaches and exceeds 0.5, complicated flow patterns that are not represented in a simple model as discussed.

3.2 Computational fluid dynamics

In this chapter, an introduction and fundamentals of computational fluid dynamics (CFD) is presented. The open source CFD code OpenFOAM will be presented. This chapter will also include a theory part where the governing equations, discretization, solution algorithms and turbulence modeling that are used in this thesis is presented, residual control is also discussed. In the end, the actuator line model (ALM) and the extension library *turbinesFoam* that is used in this work is presented. For a more thorough introduction regarding CFD, references are made to e.g. [34-36]

3.2.1 CFD introduction

CFD is becoming more common as a fundamental approach to design and analysis of engineering systems that involves fluid flow. There are many CFD codes that are being used today, there are commercial ones such as CFX/Ansys and Fluent, then there are open source codes like OpenFOAM, the latter have been used in this thesis. The CFD codes contain three main elements; a pre-processor, a solver and a post-processor. An example is presented to get a better understanding of how the solution procedure is when using CFD.

Pre-processor

The size of the computational domain is specified along with the grid (or mesh) generation. The grid generation divides the domain into individual elements, also referred to as cells, where the domain can be considered either as two-dimensional (2D) or three-dimensional (3D) domain. For 2D and 3D domains, each cell has an area surface or contains a volume, respectively. Each cell is considered as a control volume (CV) in which discretized versions of conservation equations are solved. Conservation- and transport-equations are presented later in section 3.2.2.

The boundary conditions are specified at the edges (2D) or faces (3D) of the computational domain as shown in Figure 10. There are two main types of boundary conditions: inlet/outlet and wall. These boundary conditions should be specified, in addition to the fluid properties, in the CFD model.

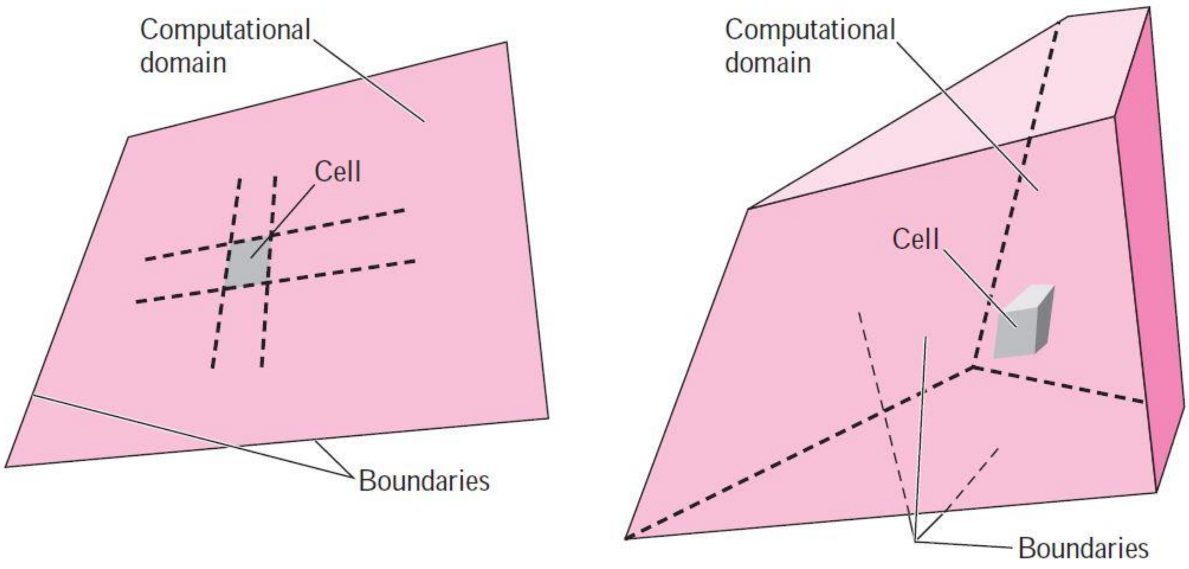


Figure 10. Computational domain, two- and three-dimensional domain is shown respectively. [36, pp. 819]

The last step in the pre-processor is to select the solution algorithms and numerical parameters. The solution algorithm used in this thesis will be presented in section 3.2.6.

Solver

The solver computes the solution which has been defined in the pre-processor. Several numerical techniques can be applied in CFD analysis; finite difference, finite element and spectral methods. The most established codes are based on finite volume method, which is a special finite difference formulation. OpenFOAM uses finite volume method (FVM).

The numerical algorithm consists of these following steps [35, pp. 3]:

- Integration of the governing equations of fluid flow over all the (finite) control volumes of the domain
- Discretisation – conversion of the resulting integral equation into a system of algebraic equations
- Solution of the algebraic equations by an iterative method

In other terms; the starting values for all flow field variables are specified for each cell, this is called initial conditions. With the initial guesses, discretized forms of the conservation- and transport- equation are iteratively solved. When the equations are solved, the solution should be equal to zero if every term in Navier-Stokes equation were to be put on one side of the equation. In CFD the this is never the case, and the resulting number is called a residual. This residual is a number on how much the result deviates from the exact. Once a simulation is running it is possible to monitor the residuals, which hopefully decreases by each iteration, to check if the solution/simulation converges or not.

When the solution/simulation has converged, global properties of the flow fields, such as pressure drop, and integral properties, such as forces and moments, should settle down to constant values. After convergence is reached it is possible to calculate e.g. lift and drag, forces, and moments acting on e.g. an airfoil.

Post-processor

The post-processor is used for visualization of data. The data can be presented by vector plots, 2D and 3D surface plots, particle tracking and much more. There is also the ability to make animations for dynamic result display, the author has a predilection for animated streamlines. Options for exporting data is also available.

3.2.2 Governing equations

The conservation laws involving a fluid flow and related transfer phenomena can be formulated following Eulerian or Lagrangian approach [34, pp. 45]. To describe the several physical quantities such as mass, momentum and energy, the Navier-Stokes equations can be used. The principle of conservation of mass (continuity equation) requires a mass sources and sinks and a region that will conserve its mass on a local level [34, pp. 47]. A brief description of how Reynolds Transport Theorem is used in the conservation of mass (continuity equation) for an incompressible flow;

$$\left(\frac{dB}{dt}\right)_{MV} = \int_V \left[\frac{\partial}{\partial t}(\rho b) + \nabla \cdot (\rho \mathbf{v} b)\right] dV \qquad \textit{Equation 3.25}$$

Where the B is any property of fluid (mass, momentum, energy), b the intensive value of B, MV is the material volume, V is the control volume, ρ the fluid density and $\mathbf{v}$ is the fluid velocity. When used in the continuity equation for mass, $B = m$ and $b = 1$, also the incompressibility assumption indicates that ρ does not change with the flow, $D\rho/Dt = 0$, therefore;

$$\nabla \cdot \mathbf{v} = 0$$ *Equation 3.26*

Equation 3.63, if presented in integral form, states that for incompressible flows the net flow across any control volume is zero i.e., flow in equals flow out.

The next step is to introduce the conservation of momentum equation for Newtonian fluids. The stress tensor $\boldsymbol{\tau}$ needs to be related to the flow variables to proceed further with the momentum equations;

$$\boldsymbol{\tau} = \mu\{\nabla \mathbf{v} + (\nabla \mathbf{v})^{\mathrm{T}}\} + \lambda_b(\nabla \cdot \mathbf{v})\boldsymbol{I}$$ *Equation 3.27*

Where μ is the molecular viscosity coefficient, λ_b the bulk viscosity coefficient and $\boldsymbol{I}$ is the identity tensor. The stress tensor is diverged and substituted into the general conservative form of the momentum equation, including the assumption of an incompressible flow and a constant viscosity then results in a simplified form of the momentum equation;

$$\frac{\partial}{\partial t}[\rho \mathbf{v}] + \nabla \cdot \{\rho \mathbf{v}\mathbf{v}\} = -\nabla p + \mu \nabla^2 \mathbf{v} + \boldsymbol{f}_b$$ *Equation 3.28*

where $\rho\mathbf{vv}$ is a dyadic product, p is pressure and $\boldsymbol{f}_b$ the body forces. For a more detailed derivation of the equations described above see [34, pp. 47-57].

The equations described above cannot be solved directly, therefore discretization and approximations must be done. This will be further discussed in 3.2.5.

3.2.3 OpenFOAM

Open Source Field Operation and Manipulation (OpenFOAM) is an open source C++ library. OpenFOAM uses executables known as applications. The applications are split into two categories, solvers and utilities. The solvers are designed to solve specific problems in continuum mechanics, and the utilities are used for data manipulations. The utilities are used in the pre- and post-processors. Users that are capable in C++ and have knowledge of the method and physics, can create their own solvers and utilities [37, pp. U-13].

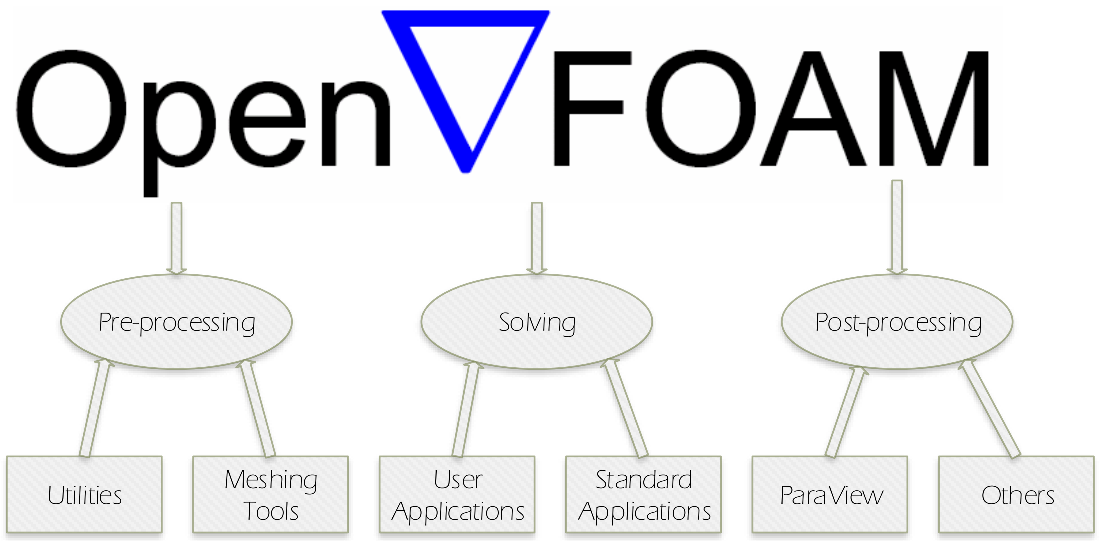


Figure 11. Overview of OpenFOAM structure. Inspired by Figure 1.1 [37]

The work done in this thesis was presented on The Fifth Symposium on OpenFOAM in Wind Energy in April 2017 [38]. The author got the impression from the other contestants, which was mostly researchers that uses OpenFOAM, that the open source software exceeds and will continue to grow beyond other CFD codes that are being developed in-house and are confidential.

OpenFOAM case setup

OpenFOAM requires a minimum set of files to be able to run an application. The files are put in the case directory and consists of; *constant*, *system* and a *0/time* directory. The case directory structure is shown in Figure 12.

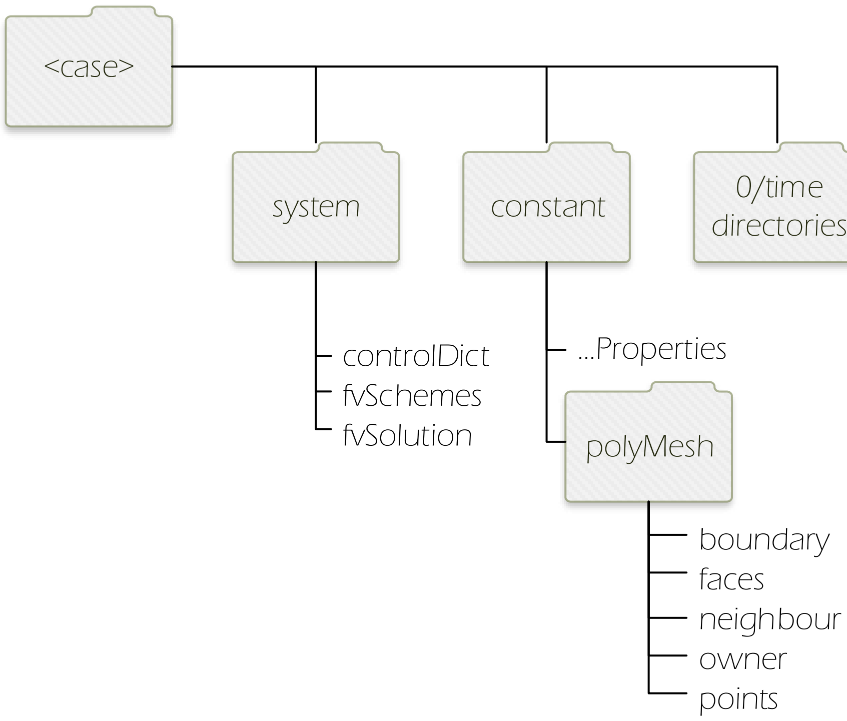


Figure 12. Case directory structure. Inspired by Figure 2.1 [37]

The different case directories contain information about;

The *system* directory, have all the parameters needed for the solution procedure itself. The system directory needs at least three files to run, a controlDict file where run control parameters are set, fvSchemes where the discretisation schemes are selected, and a fvSolution where the equation solvers, tolerances and other algorithm controls are set.

The *constant* directory, have a description of the mesh in a sub-directory called polyMesh, it also contains files about the physical properties for the application that is being used.

The *time* directory, contains data for each individual field. Often the time directory that contains all the initial field- and boundary conditions are called "0". As the simulation runs other time directories are outputted, after what has been specified in the controlDict file, which contains data of the solved fields.[1]

[1] As a tip to whomever may read this, a "0.org" file should be created and be copied to a "0" directory each time one run a simulation so the original 0 directory are not affected by the simulation, which in some cases can be manipulated by the solver.

3.2.4 Turbulence modeling

This chapter will introduce turbulent modeling and turbulent flow simulation methods. Then Reynolds averaging and the k-ϵ turbulence model will be discussed in detail since it is used in this project.

Turbulent Flow

An important variable when describing turbulence flow is the Reynolds number, which is already introduced in Equation 3.5. Turbulent flow is chaotic and time-dependent and with rapid mixing, and there are three-dimensional vorticity fluctuations with different time- and length scales. When flow reaches a certain Reynolds number laminar flow starts developing into a more turbulent flow [34, pp. 693].

Kolmogorov developed a concept based on energy cascade to describe the theory of turbulence. This theory states that, turbulence is composed of eddies of varied sizes and containing energy depending on the dimension of it. The larger eddies dissolves into smaller eddies until they are so small that they dissipate the turbulent kinetic energy as heat.

Turbulent flow is complex and difficult to model. There are ways to simulate using different models, which can be categorized into three main categories [34, pp. 694-5]:

Direct numerical simulation (DNS)

The Navier-Stokes equations for turbulent flows are solved directly. This requires a fine mesh where the length of one cell must be smaller than the smallest eddies on the Kolmogorov micro length scale. The timestep also must be smaller than the Kolmogorov time scales, and is limited by a Courant number below one. Simulating at a such a fine resolution have a high computational cost.

Large eddy simulation (LES)

A less computational intensive model that DNS has been devised. Large eddy simulation is such a method. This method uses a spatial statistical filter on the Navier-Stokes equation to determine which eddies to keep and which to model. The larger eddies that are kept, are directly simulated and the small turbulent scales are modeled using a sub-grid scale model.

Reynolds-averaged Navier-Stokes equations (RANS)

This approach decomposes the flow variables into a time averaged component and a fluctuation one. RANS focus on the mean value, and how turbulent fluctuations affects the mean value. This is the most popular approach by the industry for solving turbulent flow problems, mainly because of it is less computational demanding.

As mentioned above RANS is used in this project so a more detailed explanation follows in the next sub-chapters. For a fully detailed review see [34, pp. 693-744, 39]

Reynolds-Averaged Navier-Stokes equations

Reynolds averaging is used to find the mean value of a turbulent flow property. The instantaneous value of a flow variable can be generally explained as:

$$\phi(\boldsymbol{x},t) = \bar{\phi}(\boldsymbol{x},t) + \phi'(\boldsymbol{x},t) \qquad \text{Equation 3.29}$$

Where ϕ is representing a flow variables such as velocity or pressure, at the time t and position $\boldsymbol{x}$. $\bar{\phi}(\boldsymbol{x},t)$ is a decomposed mean value component and $\phi'(\boldsymbol{x},t)$ is a fluctuating component. A graphic representation of the components is presented in Figure 13.

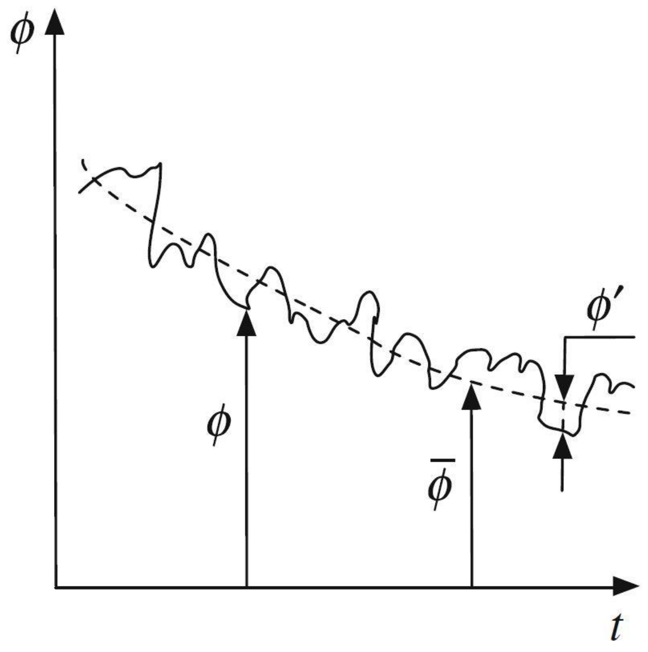


Figure 13. Fluctuating and mean variable components. [34, pp. 695]

This is called Reynolds decomposition. The decomposed variables for velocity and pressure are inserted into Navier-Stokes equations, continuity equation (Equation 3.26) and the momentum equation (Equation 3.28). The Reynolds averaged forms these equations are obtained as:

$$\nabla \cdot [\rho \overline{\mathbf{v}}] = 0$$ Equation 3.30

$$\frac{\partial}{\partial t}[\rho \overline{\mathbf{v}}] + \nabla \cdot \{\rho \overline{\mathbf{v}}\overline{\mathbf{v}}\} = -\nabla \overline{p} + \left[\nabla\left(\overline{\boldsymbol{\tau}} - \rho \overline{\mathbf{v}'\mathbf{v}'}\right)\right] + \overline{\boldsymbol{f}}_b$$ Equation 3.31

The equations above are similar to the original Navier-Stokes equations with the exception being the additional averaged products of the fluctuating components. This introduces introduced six new unknows to the momentum equation, which is the components of the tensor $\rho\overline{\mathbf{v}'\mathbf{v}'}$, this is known as the Reynolds stress tensor $\boldsymbol{\tau}^R$:

$$\boldsymbol{\tau}^R = -\rho \begin{pmatrix} \overline{u'u'} & \overline{u'v'} & \overline{u'w'} \\ \overline{u'v'} & \overline{v'v'} & \overline{v'w'} \\ \overline{u'w'} & \overline{v'w'} & \overline{w'w'} \end{pmatrix}$$ Equation 3.32

The set of RANS equations are not closed, to solve it additional equations are need for the Reynolds stress components. The Reynolds stress tensor comes from a non-linear convection term and any attempt to linear averaging the equations, as done in the Reynolds averaging, cannot reduce the order of the problem. It is only getting more and more complex. To overcome this, turbulence models must close the system of equations, this is called the closure problem.

There are many ways to model the Reynolds stresses and the eddy viscosity, and many of them are based on the Boussinesq hypothesis. Boussinesq hypothesis are used to directly model the Reynolds stress tensor, and for incompressible flow can be reduced to:

$$\boldsymbol{\tau}^R = -\rho\overline{\mathbf{v}'\mathbf{v}'} = \mu_t\{\nabla\mathbf{v} + (\nabla\mathbf{v})^{\mathrm{T}}\} - \frac{2}{3}\rho k \boldsymbol{I}$$ Equation 3.33

This modeling of the Reynolds stress tensors assumes that the Reynolds stress are a linear function of the mean velocity gradients.

As mentioned above there are several turbulence models based on Boussinesq hypothesis that has been developed to calculate the turbulent viscosity μ_t, in terms of velocity $\sqrt{k}$ and length scale l:

$$\mu_t = \rho l \sqrt{k}$$ Equation 3.34

The four main groups of turbulence models are as follows:

- Algebraic (Zero-Equation) Models
- One-Equation Models
- Two-Equation Models
- Second-Order Closure Models

The main difference of the groups is the amount of additional transport equations that are solved. As the name reveals the zero-equation models does not need an additional transport equation, e.g. *the mixing length model*. The one-equation model requires uses one transport differential equation to compute the turbulent eddy viscosity, e.g. *Spalart-Allmaras model*. The two-equation models require two transport equations to calculate the turbulence viscosity, e.g. *k-ϵ model*. The last group are the second-order closure models, and they need to solve a transport equation for each of the turbulent fluxes and one in addition for the dissipation rate of kinetic energy, all these result in a total of seven transport equations, e.g. *the Reynolds stress model*.

The different models mentioned above each have their advantages and none is applicable for all flow conditions. The more transport equations one solves the more accuracy is gained, but at computational cost. This is the main reason the two-equation models have gained the more attention from the industry, it is a good combination of accuracy and computational cost. In this project, the $k - \epsilon$ model has been used and will be further discussed in the next section.

Standard $\boldsymbol{k - \epsilon}$ Model

The $k - \epsilon$ model solves one transport equation for the turbulent kinetic energy and one for the dissipation of turbulent kinetic energy. With these variables, an approximation to the eddy viscosity and the Reynolds stresses can be made.

The turbulent kinetic energy is defined as:

$$k = \frac{1}{2}\overline{\mathbf{v}' \cdot \mathbf{v}'}$$ *Equation 3.35*

and the rate of dissipation of turbulence kinetic energy per unit mass due to viscous stresses given by:

$$\epsilon = \frac{1}{2}\frac{\mu}{\rho}\overline{\{\nabla\boldsymbol{v}' + (\nabla\boldsymbol{v}')^T\} : \{\nabla\boldsymbol{v}' + (\nabla\boldsymbol{v}')^T\}}$$ *Equation 3.36*

The eddy viscosity is defined as:

$$\mu_t = \rho C_\mu \frac{k^2}{\epsilon} \qquad \text{Equation 3.37}$$

where C_μ is a dimensionless constant.

The turbulent kinetic energy and the turbulent energy dissipation rate are computed using the differential equations deduced from the Navier-Stokes equations, model equation for k:

$$\underbrace{\frac{\partial}{\partial t}(\rho k)}_{Transient} + \underbrace{\nabla \cdot (\rho \boldsymbol{v} k)}_{Convective} = \underbrace{\nabla \cdot \left(\mu_{eff,k} \nabla k\right)}_{Transport} + \underbrace{P_k}_{Production} - \underbrace{\rho \epsilon}_{Dissipation} \qquad \text{Equation 3.38}$$

Then for ϵ:

$$\underbrace{\frac{\partial}{\partial t}(\rho \epsilon)}_{Transient} + \underbrace{\nabla \cdot (\rho \boldsymbol{v} \epsilon)}_{Convective} = \underbrace{\nabla \cdot \left(\mu_{eff,\epsilon} \nabla \epsilon\right)}_{Transport} + \underbrace{C_{\epsilon 1} \frac{\epsilon}{k} P_k}_{Production} - \underbrace{C_{\epsilon 2}\, \rho \frac{\epsilon^2}{k}}_{Dissipation} \qquad \text{Equation 3.39}$$

Where,

$$\mu_{eff,k} = \mu + \frac{\mu_t}{\sigma_k}, \qquad \mu_{eff,\epsilon} = \mu + \frac{\mu_t}{\sigma_\epsilon} \qquad \text{Equation 3.40}$$

with the constant assigned to: $C_\mu = 0.09$, $\sigma_k = 1.00$, $\sigma_\epsilon = 1.30$, $C_{\epsilon 1} = 1.44$, $C_{\epsilon 2} = 1.92$. A compact form of the production of turbulent energy term is used, $P_k = \boldsymbol{\tau}^R : \nabla \boldsymbol{v}$.

From the last term in Equation 3.38 one can see that the destruction of k is proportional to ϵ, which makes sense because the two are closely linked where k is the creation of turbulent energy and ϵ stands for its destruction.

The $k - \epsilon$ model is only valid for fully developed turbulent flows, which is not the case when the flow is close to the wall. This therefore must be modeled and will be discussed in the next section.

Wall boundary layer

The boundary layers close to the wall is divided into regions. Depending on the value of Reynolds number different flow fields appear, as shown in Figure 14:

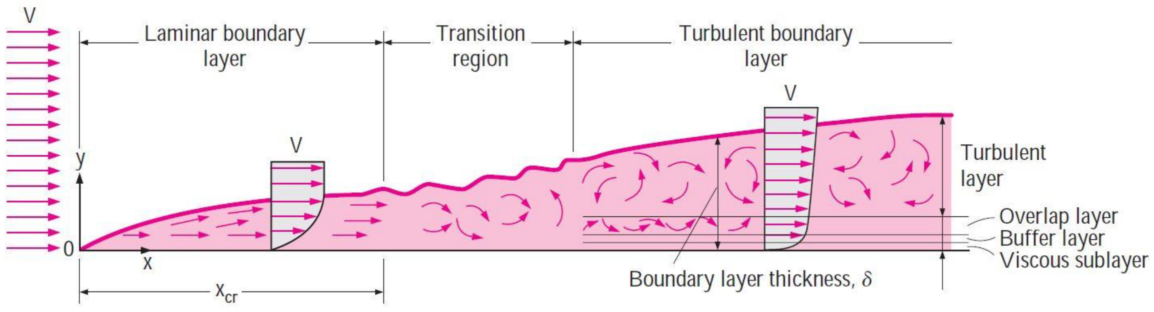


Figure 14. Schematic of the flow over a flat plate showing the laminar, transitional, and turbulent flow regimes based on the value of Reynolds number. Not to scale. [36, pp. 579]

As mentioned the $k - \epsilon$ turbulence model is only valid for fully developed turbulent flows, and when the flow is as in the turbulent flow region as seen in Figure 14, theoretical profiles between the boundary surface and the first near-wall node are assumed and superimposed with so called *wall functions*.

The boundary layer is divided into an inner region, where viscous shear dominates and the shear stress is approximately equal to the wall stress, and an outer region where turbulent shear stresses dominates.

The inner region of the flow can be divided into three regions; viscous sublayer ($0 < d^+ < 5$), the buffer sublayer ($5 < d^+ < 30$), and the inertial sublayer[2] ($30 < d^+ < 300$), where d^+ is the normalized distance to the wall:

$$d^+ = \frac{d_\perp u_\tau}{v} = y^+ \qquad \text{Equation 3.41}$$

Where $d_\perp$ is the normal distance to the wall, v is the kinematic viscosity, and u_τ is the friction

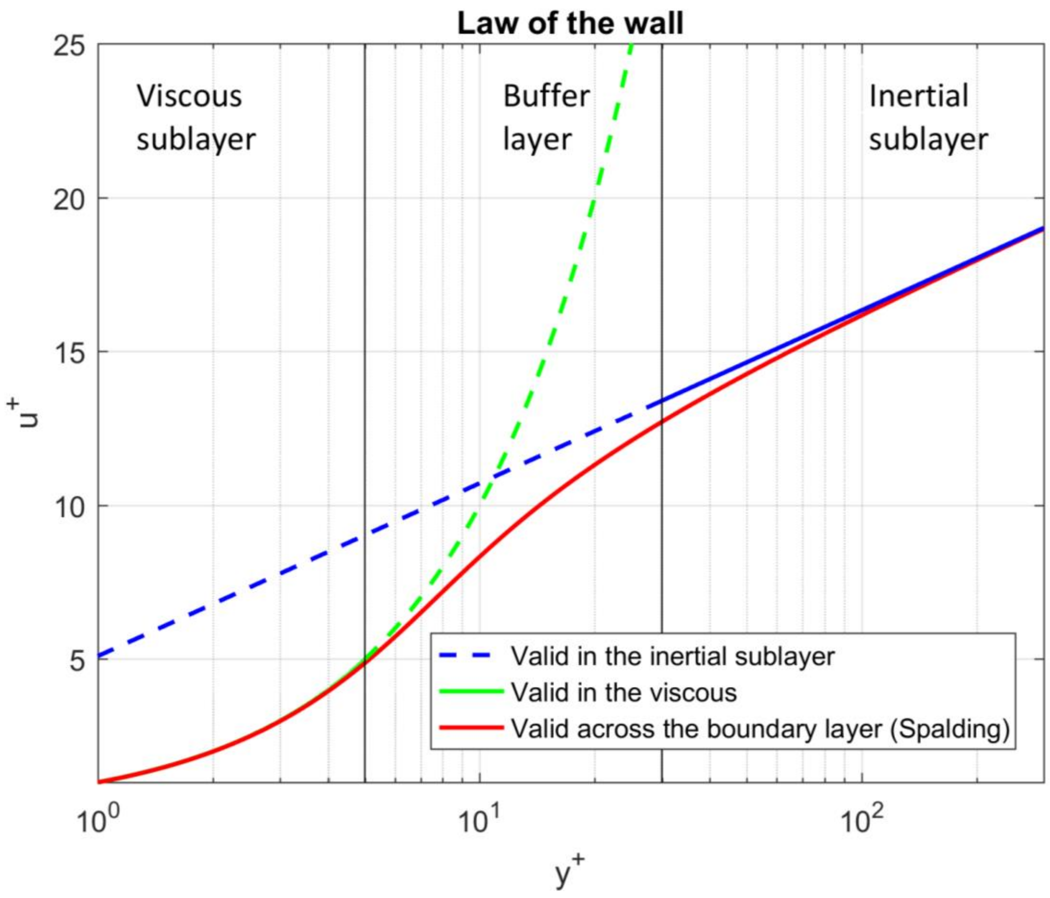


Figure 15. Velocity distribution in a boundary layer, green line is valid in the viscous sublayer $y^+ < 5$, blue line is valid in the inertial sublayery$^+ > 30$, and the red line is valid across the boundary layer.

velocity expressed in terms of the wall shear stress τ_w as:

$$u_\tau = \sqrt{\frac{|\tau_w|}{\rho}} \qquad \text{Equation 3.42}$$

[2] Also known as the log-law region or fully turbulent region

where $|\tau_w|$ is the magnitude of the wall shear stress. The relationship between the velocity and the distance (see Figure 15) from the wall is given;

- Viscous sublayer, linear relationship between the velocity and the distance from the wall, $u^+ = d^+$. Valid where the green line is solid in Figure 15.
- Buffer layer, no defined relations between u^+ and d^+.
- Inertial sublayer, the relationship is defined as $u^+ = \frac{1}{\kappa}\ln(d^+) + B$, where $\kappa \approx 0.41$ is von Karmann's constant and $B \approx 5.25$. Valid where the blue line is solid in Figure 15.

Spalding showed that experimental velocity distributions may be well fitted across the whole boundary layer, by the formula [40]:

$$d^+ = u^+ + e^{-\kappa B}\left[e^{\kappa u^+} - 1 - \kappa u^+ - \frac{(\kappa u^+)^2}{2} - \frac{(\kappa u^+)^3}{6}\right] \qquad \text{Equation 3.43}$$

The modeling of the flow in both the viscous sublayer and the inertial sublayer is "easier" than modeling the buffer region. This is due to the turbulence is negligible in the viscous sublayer, and the viscous effects are small in the inertial sublayer where as in the buffer layer both effects are important. In practice, when using the $k - \epsilon$ turbulence model avoids modeling the buffer layer near a wall by placing the first grid point in the inertial sublayer [34, pp. 711].

The theory above is based on a standard wall function, there is a wide range of different wall functions which will not be explained, the ones that are used in this project is only mentioned in 5.3.1.

3.2.5 Discretization

This chapter will only introduce the different steps of the discretization process with the use of the finite volume method in OpenFOAM, which include; the modeling of the domain (meshing), numerical discretization that transforms the governed equations to a set of algebraic equations defined on each of the elements of the computational domain and the solution method used to solve the system of algebraic equations. For a fully detailed review of the discretization process, please see [34].

The discretisation process can illustrated as Figure 16:

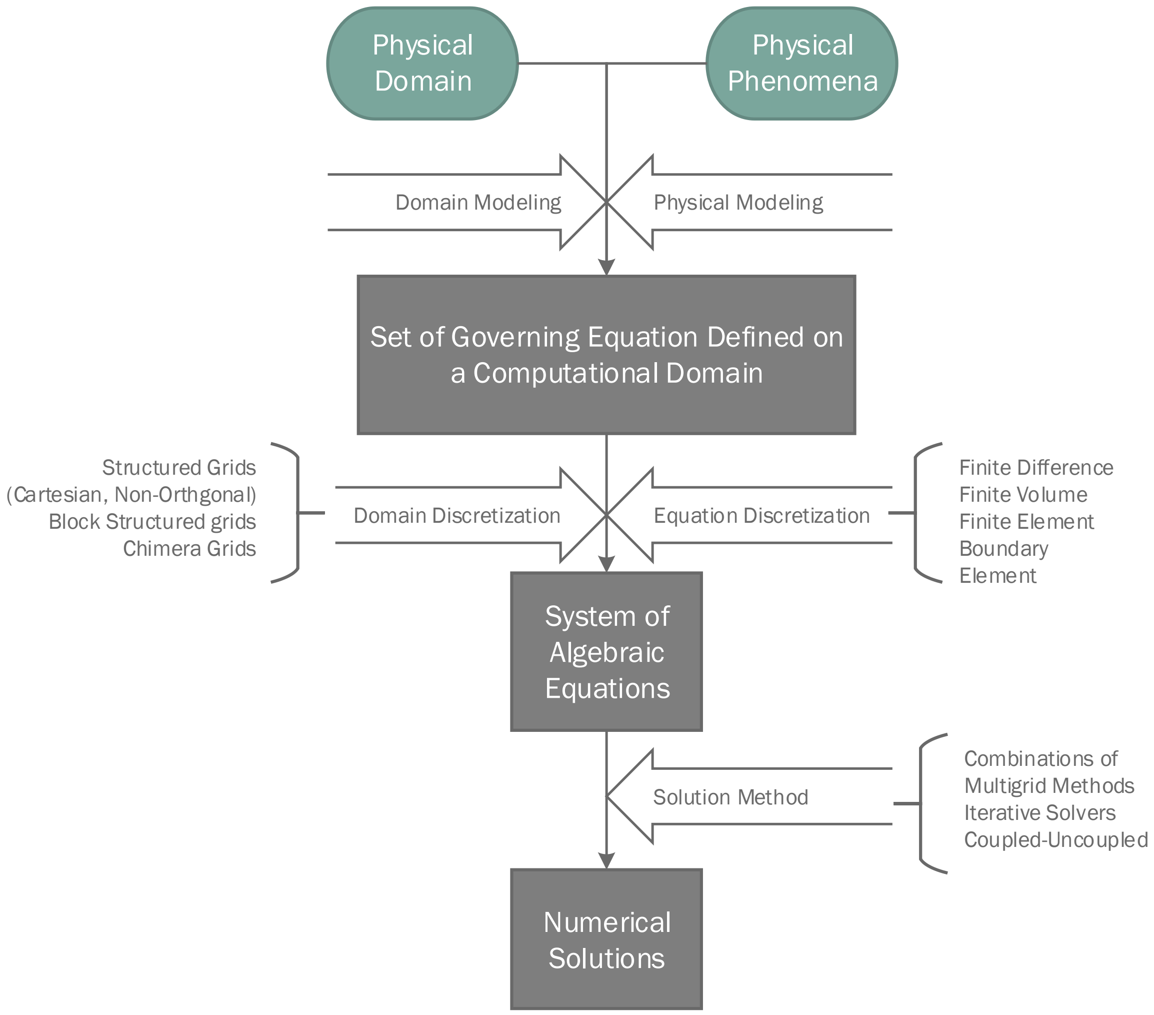
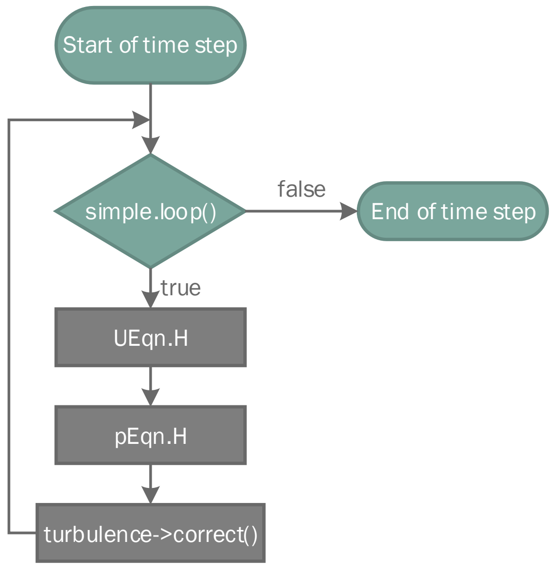


Figure 16. The discretization process. Inspired by [34, pp. 86]

3.2.6 Solution Algorithms

Once the discretization is process done, the system of equations must be solved using linear algebra. OpenFOAM uses algorithms that decouple the pressure from velocity to simplify the task. Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) and Pressure Implicit with Splitting of Operators (PISO) are algorithms that contain procedures for solving pressure-velocity coupled equations. The SIMPLE algorithm and PISO are used for steady-state problems and transient ones, respectively [37]. The solution algorithm that is used in this thesis is a combination of these to solvers, named PIMPLE. The PIMPLE solution algorithm can be used for both steady-state and transient problems. A brief review of the algorithms will be presented.

The SIMPLE algorithm computed the velocity and then adjusts pressure and velocity in a feedback loop until convergence criteria is reached, see Figure 17a. The PISO algorithm has a different order on the procedure but it follows the same steps. The velocity is predicted using the momentum predictor first, and only then the pressure and velocity are corrected until the predefined numbers of iterations is reached, see Figure 17b.



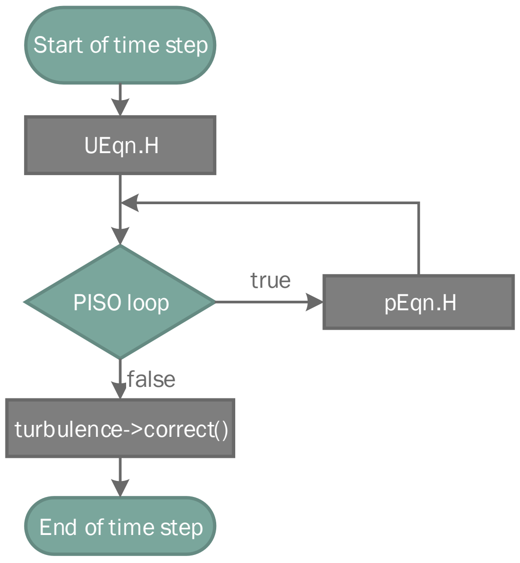


Figure 17. Flow chart for SIMPLE (a) and PISO (b) algorithms in OpenFOAM, respectively. Inspired by [37]

The momentum predictor is denoted by *UEqn.H* file where *pEqn.H* will correct the pressure and velocities.

The PIMPLE algorithm is like PISO, but contains an extra loop as shown in Figure 18. One thing to notice is that there is a slight difference for the turbulence approach. In the step *turbCorr()*, OpenFOAM decides whether to tackle turbulence or not depending on the simulation type. For laminar cases, the turbulence correction is skipped [37].

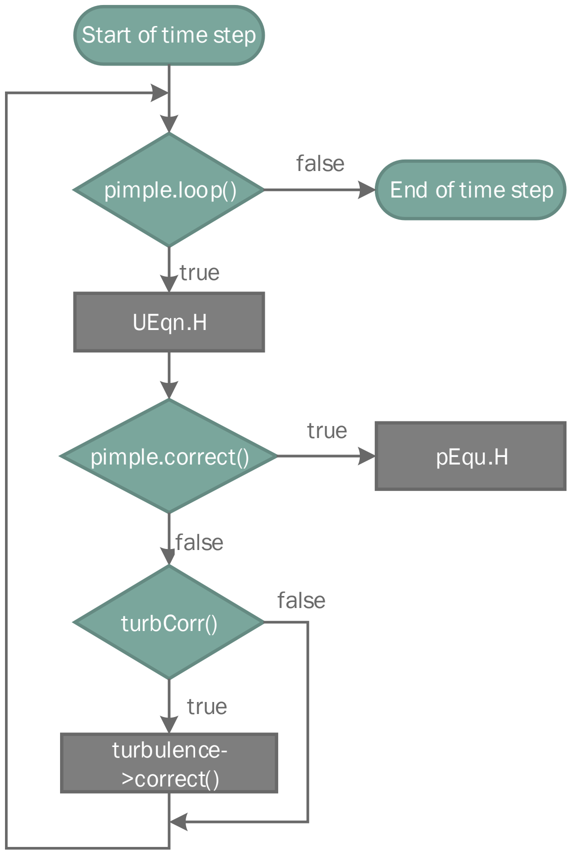


Figure 18. PIMPLE algorithm flow chart. Inspired by [37]

3.2.7 Residual control

When the numerical simulation is run it is important to make sure that the convergence of the solution is reached. A solution that is converged is a good indicator to identify if the simulation is physically correct or not. The best way to check if the simulation is physically correct or not is to validate the result with experimental data. One can monitor the residuals and continuity errors while the simulation is running with e.g. tailing the log file (if made) and check for continuity errors, plot the residuals with *gnuplot* using a script or use OpenFOAM's built in function script *foamMonitor* or *foamLog.*

Residuals are the imbalance, or error, that occur in the equations for each solved variable. The lower the residual the more accurate the solution will be. The residuals can be calculated by substituting the current solution for a time step into the equations and taking the absolute value of the left and right-hand side. To make the result independent of the scale of the problem the residual is normalized [41].

The Courant number is defined as:

$$Co = \frac{\Delta t \cdot |U|}{\Delta x}$$ *Equation 3.44*

where Δt is the length of the time step, U is the flow velocity and Δx is the length of the cell. The Courant number represents the portion of a cell that a solute will traverse by advection in one time step. When advection dominates dispersion, a smaller Courant number will decrease oscillations, improve accuracy and decrease numerical dispersion. The Courant number should be kept under one [42].

3.2.8 Atmospheric Boundary Layer

The model has been enriched with an atmospheric boundary layer. *OpenFOAM* has a build in functions for this called *atmBoundaryLayer* where one can include the boundary conditions. The reference paper that this function was built on was Hargreaves and Wright, which studied the use of commercial CFD software to model the atmospheric boundary layer (ABL) [43]. The result was a class that provides functions to evaluate the velocity and turbulence distributions appropriate for ALB. The profile is derived from the friction velocity, flow direction and "vertical" direction as [44]:

$$U = \frac{U^*}{\kappa} \ln\left(\frac{z - z_g + z_0}{z_0}\right)$$ Equation 3.45

$$k = \frac{(U^*)^2}{\sqrt{C_\mu}}$$ Equation 3.46

$$\epsilon = \frac{(U^*)^3}{\kappa\left(z - z_g + z_0\right)}$$ Equation 3.47

where U^* is the friction velocity, z is the vertical coordinate, z_0 is the surface roughness length and z_g is the minimum z-coordinate. The friction velocity is defined as:

$$U^* = \kappa \frac{U_{ref}}{\ln\left(\frac{z_{ref} + z_0}{z_0}\right)}$$ Equation 3.48

where U_{ref} is the reference velocity at Z_{ref}, which is the reference height.

These formulas for the turbulent kinetic energy and dissipation have not been used. They have been calculated as discussed previously, with the use of the characteristics length of the ALM model. There will therefore preformed horizontally homogenous turbulent surface layer analysis.

A comparison between the wind profiles in the simulations and De Norske Veritas (DNV) standard DNV-OS-J101 [45] suggestion for the logarithmic wind speed profile for neutral atmospheric conditions, which is given by:

$$U(z) = \left(\frac{u_*}{\kappa}\right) \ln\left(\frac{z}{z_0}\right)$$ Equation 3.49

Where z_0 is the roughness length calculated using the Charnock relation and u_* is the friction velocity given by [46, 2.3.2.3]. The friction velocity can be calculated using the surface friction coefficient is given by [46, 2.3.2.6]:

$$u_* = K^{\frac{1}{2}} * z_{ref}$$ Equation 3.50

$$K = \left(\frac{\kappa^2}{\log\left(\frac{z_{ref}}{z_0}\right)} \right)^2$$ Equation 3.51

After discussion with Kalvig, it has later on been decided that the roughness length that was used for the simulations was a bit high, a rerun was performed with the $z_0 = 0.002m$, which represents calm open sea from [18, pp. 46] instead of $0.01m$ from [46], which is open sea with waves.

There was not found any significant difference in the velocity field, this is probably because the impact of the roughness length is negligible compared to impact of the implemented geometry. A rerun should be done in an empty grid, due to time restriction this was not done.

3.3 Actuator Line Model

The actuator line model (ALM) that is being used has been adopted from the reference paper [5]. In this section, a summary of the theory that is involved will be presented, for a full review please see [5]. This theory is implemented by a *OpenFOAM* library *turbinesFoam* [4], which is a ALM for both the horizontal- and vertical axis wind turbine, which is dubbed axial- and cross flow turbines, respectively. The source file has been inspected and a flow chart of how the model works will be presented. At last the output of the ALM and how this is being used in the result will be presented.

3.3.1 The Actuator Line Model Theory

The actuator line model is based on the classical blade element theory combined with a Navier-Stokes description of the flow field. The model treats the blades, struts and shaft of the turbine as actuator line elements, which is defined by the quarter-cord location. At the quarter-cord location the inflow velocity, $\vec{U}_{in}$, is sampled field with the use of *OpenFOAM's interpolationCellPoint*. With the inflow velocity and relative blade motion $-\omega r$, where ω is the rotor angular velocity and r is the blade element radius, the relative flow velocity $\vec{U}_{rel}$ and the angle of attack α are computed. With the know location of the quarter cord, the ALM imports 2-D blade profile lift and drag coefficients from a predefined table. The vector diagram of the ALM and the turbine used, UNH-RVAT, is shown in Figure 19:

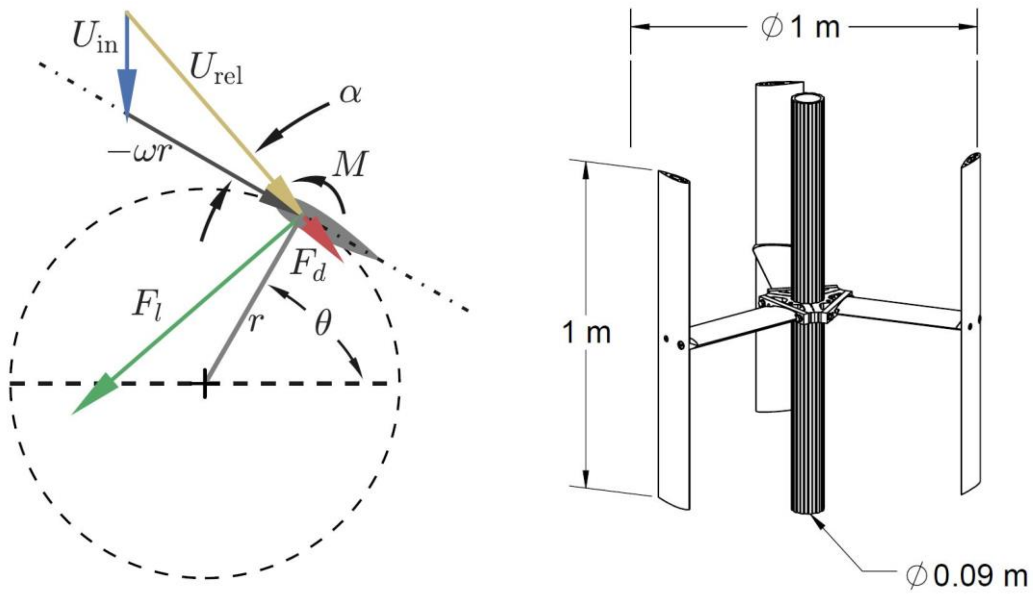


Figure 19. Vector diagram for velocity and forces of the ALM and a drawing of the turbine used, UNH-RVAT, respectively. The vectors are not in scale, they are magnified to enhance visibility. [5]

The kinematics of a vertical axis wind turbine are parameterized by the tip speed ratio:

$$\lambda = \frac{\omega R}{U_\infty} \qquad \text{Equation 3.52}$$

Where the R is the maximum rotor radius and U_∞ is the free stream velocity. The tip speed ratio has been kept at 1.9 throughout this thesis.

The model is strictly affected by unsteadiness. The unsteadiness can be characterized by a reduced frequency[3]:

$$k_{rf} = \frac{\lambda c}{2R} \qquad \text{Equation 3.53}$$

where the c/R represents solidity. Unsteady effects become significant for $k_{rf} > 0.05$ and can become dominant for $k_{rf} \geq 0.2$. For smaller turbines, the solidity is often higher, which in this case it is, so to handle these unsteady effects they need to be modeled. The unsteady effects; dynamic stall and added mass, together with other corrections such as flow curvature, and end effects have not been considered in this thesis, the correctors have been kept on and the default values has been used.

Assuming the unsteady effects can be appropriately modeled, the lift- and drag force and the pitching moment are calculated as

$$F_l = \frac{1}{2}\rho A_{elem} C_l \left|\vec{U}_{rel}\right|^2 \qquad \text{Equation 3.54}$$

$$F_d = \frac{1}{2}\rho A_{elem} C_d \left|\vec{U}_{rel}\right|^2 \qquad \text{Equation 3.55}$$

$$M = \frac{1}{2}\rho A_{elem} c C_m \left|\vec{U}_{rel}\right|^2 \qquad \text{Equation 3.56}$$

respectively, where ρ is the fluid density, A_{elem} is the blade platform area ($span * cord$), $\vec{U}_{rel}$ is the local relative velocity projected onto the plane of the element cross section. The $C_l, C_d\ and\ C_m$ are sectional lift, drag and pitching moment coefficient, respectively,

[3] Added a denotation rf for the reduced frequency so that variable will not be mistaken for the turbulent kinetic energy

coefficients are interpolated form a predefined table per the local angle of attack. The forces are projected onto the rotor coordinate system to calculate torque, overall drag and lift. When the forces on the actuator lines from the flow is computed, they are added to the Naviers-Stokes equations, Equation 3.28, as body force or momentum source $\boldsymbol{f}_b$, per unit density, assuming incompressible flow.

When the forces on the actuator line elements from the flow is calculated, it is then projected back onto the flow field as a source term in the momentum equation. To visualize it, the forces is smeared back onto grid spherically, this is done by a spherical Gaussian function, where the only variable is the Gaussian width parameter ϵ_{mesh}. The variable used is the largest of these three options, one relative to the chord length, one to the mesh size, and one to the momentum thickness du to drag force. According to the reference paper [5], the Gaussian width due to mesh size by estimating the size of the cell containing the element as:

$$\Delta x \approx \sqrt[3]{V_{cell}} \qquad \text{Equation 3.57}$$

where the V_{cell} is the cell volume. The formula for the Gaussian width due to mesh size is given by:

$$\epsilon_{mesh} = 2C_{mesh}\Delta x \qquad \text{Equation 3.58}$$

where the factor C_{mesh} is introduced to account for non-unity aspect ratio cells, which by trial and error was determined to set as 2.0 by the reference paper [5].

The number of elements per actuator line was set to be approximately equal to the total span divided by the Gaussian width, which gives:

$$N_{elem} = \frac{L_{span}}{\epsilon_{mesh}} \qquad \text{Equation 3.59}$$

The number of time steps should be set according to the size of the cell. The ALM should not rotate more than one cell length at the time. This has been adopted, but it does not account that the blade moves in a circular motion, while the grid is cartesian. The fastest point of the ALM is the tip of the blade which, given by the linear velocity, is the same velocity as the relative blade motion of the outer elements:

$$U_{tip} = \omega R \qquad \text{Equation 3.60}$$

Then the timestep should be set as:

$$\Delta t < \frac{\Delta x}{U_{tip}} \qquad \text{Equation 3.61}$$

3.3.2 Flow chart *turbinesFoam*

The source file and theory has been reviewed and a flow chart of how the turbinesFoam works is presented in Figure 20.

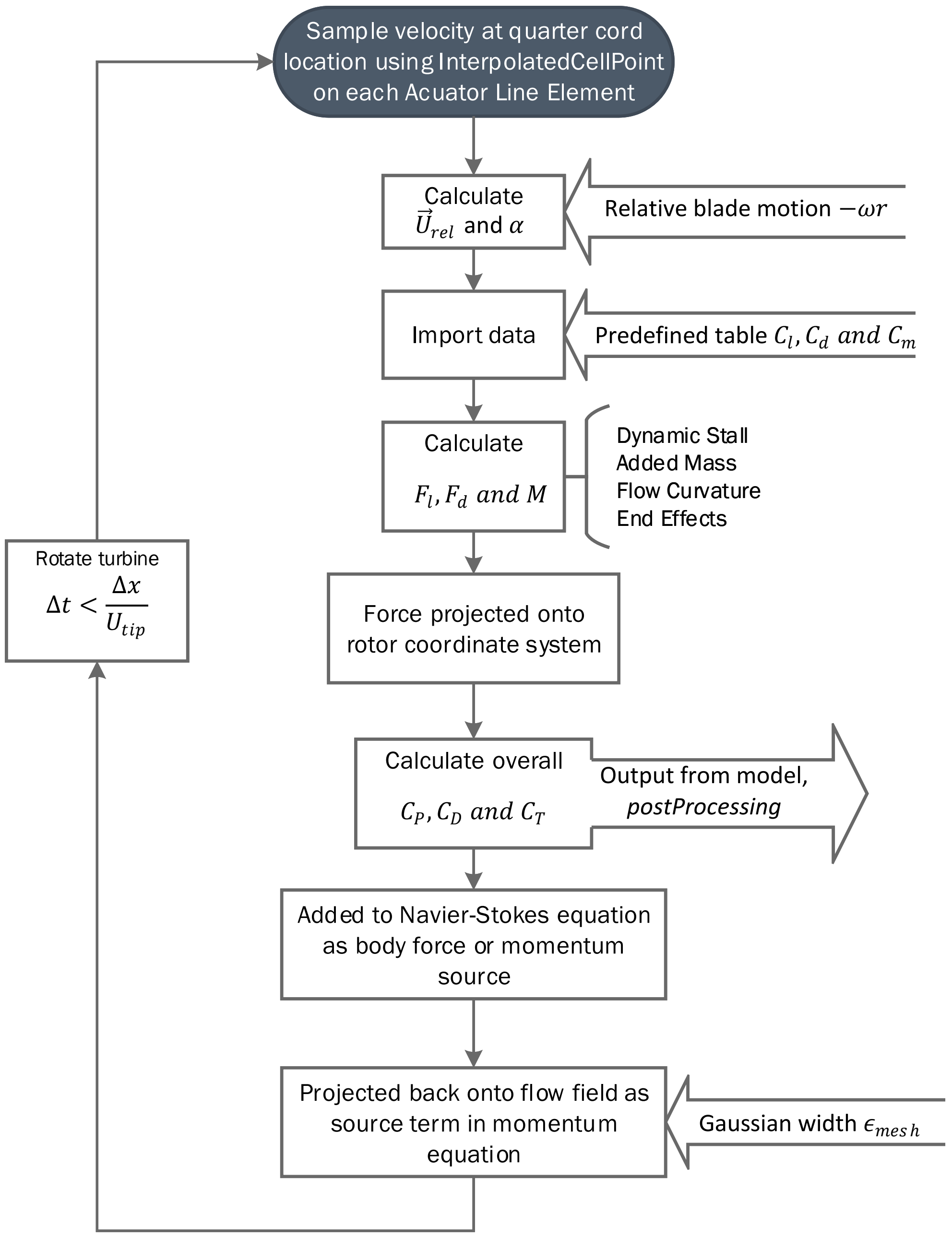

Sample velocity at quarter cord location using InterpolatedCellPoint on each Acuator Line Element
Calculate $\vec{U}_{rel}$ and α
Relative blade motion $-\omega r$
Import data
Predefined table C_l, C_d and C_m
Calculate F_l, F_d and M
Dynamic Stall
Added Mass
Flow Curvature
End Effects
Rotate turbine $\Delta t < \frac{\Delta x}{U_{tip}}$
Force projected onto rotor coordinate system
Calculate overall C_P, C_D and C_T
Output from model, *postProcessing*
Added to Navier-Stokes equation as body force or momentum source
Projected back onto flow field as source term in momentum equation
Gaussian width ϵ_{mesh}

Figure 20. Flow chart turbinesFoam. Based on source file and theory.

3.3.3 crossFlowTurbineALSoruce.C

The crossFlowTurbineALSoruce.C is the source code for the actuator line model for vertical axis wind turbines.

Protected Member Functions

The model first creates a coordinate system in the cellZone. It establishes the axis of the turbine, the free stream direction, the radial direction also calculates the azimuthal or tangential direction.

The next step is to create the blades for the actuator line model. When the number of blades is selected, the code creates a dictionary for the blades. The element data is then converted in to actuator line element geometry. Then the model imports the settings that has been put into the fvOptions to make the wind turbine as you want it. Settings such as axial distance, radius, chord length, cord mount and pitch are scaled. The settings after being scaled are then set. The frontal area of a specific geometry segment is calculated from the axial distance and mean radius see Equation 3.62.

$$frontalArea = |deltaAxial * meanRadius| \qquad \text{Equation 3.62}$$

The next step is to create a geometry origin point for the actuator line source, so that the model can rotate the wind turbine in the right direction. Then it calculates the initial velocity of the quarter cord, also called aerodynamic center, see Equation 3.63 with the corrected radius see Equation 3.64.

$$radiusCorr = \sqrt{|chordMount - 0.25|^2 * chordLength + |radius|^2} \qquad \text{Equation 3.63}$$

$$initalVelocity = -freeStreamDirection * \omega * radiusCorr \qquad \text{Equation 3.64}$$

The model rotates the aerodynamic center and initial velocity according to azimuth value. The frontal area is twice the maximum blade frontal area see Equation 3.65.

$$frontalArea = 2 * \max(frontalAreas) \qquad \text{Equation 3.65}$$

Then the model converts the element data into actuator line element geometry for the struts and shaft. They are rotated in an equivalent way as the blades.

Member functions

The model rotates if the time value has changed, then zero out force vector and field. To be able to calculate the force and moment it need to add the sources for blade-, strut- and shaft actuator lines. The torque can then be calculated using the projection of the moment from all the blades on the axis, the inner product of the moment and axis see Equation 3.66.

$$torque = \boldsymbol{moment} \cdot \boldsymbol{axis} \qquad \text{Equation 3.66}$$

With the torque, the torque coefficient can be calculated with Equation 3.67.

$$C_T = torqueCoefficient = \frac{torque}{\frac{1}{2} * frontalArea * rotorRadius * freeStreamVelocity^2} \qquad \text{Equation 3.67}$$

The power coefficient can be calculated with the tip speed ratio and torque coefficient with Equation 3.68.

$$C_P = powerCoefficient = torqueCoefficient * tipSpeedRatio \qquad \text{Equation 3.68}$$

The drag is also calculated with Equation 3.69.

$$C_D = dragCoefficient = \frac{force \cdot freeStreamDirection}{\frac{1}{2} * frontalArea * freeStreamVelocity^2} \qquad \text{Equation 3.69}$$

The performance coefficients are printed in the terminal and is also saved in an own document in the *postProcessing* folder. If the model is used with an environment with a different density the model calculates the coefficients with *rhoRef*.

3.3.4 Output *turbinesFoam*

The model outputs data from actuator lines and the performance of the turbine. The data from the actuator lines, contains information about the position, $\vec{U}_{rel}$, α and the interpolated values for C_d, C_l and C_m. The performance of the turbine is outputted, the non-dimensionalized coefficients; C_T, C_P and C_D, and the azimuthal angle is saved for each timestep. They are calculated as explained in the last section.

The important thing to notice is that *freeStreamVelocity* is used for non-dimensionalizing the performance coefficients, which is not actual wind upstream of the turbine. This project has been enriched with ABL and a geometry has been implemented, so the velocity, in this case is higher than the *freeStreamVelocity*, will lead to overprediction of the turbine performance. To correct for this one could use a sampled velocity field upstream of the turbine, then calculate a new power coefficient as:

$$C_{P,new} = C_P \frac{V_{freeStream}^3}{V_{sampled}^3} \qquad \text{Equation 3.70}$$

Initially, the plan was to change the source code so that this was corrected in the model, but due to the time-constraint and scope of this thesis, this was not performed. Pete Bachant, the author of the *turbinesFoam* library has been contacted in the process of finding a solution. He put an issue tracker regarding the development of a tip speed ratio controller, which then would use some sampled velocity upstream, but this work is a complex challenge and requires more elaborate studies and analysis [47].

Further, if nothing else is mentioned, in this thesis when mentioning the power coefficient this will refer to the C_P outputted from the model, and not the corrected one.

To avoid this problem of the nondimensionalizing, the torque has been used in the results. The torque coefficient has been used to calculate back to torque since this is not outputted from the model, as:

$$T = C_T * \frac{1}{2} * \rho_{ref} * A_{front} * R_{rotor} * V_{freeStream}^2 \qquad \text{Equation 3.71}$$

where C_T is the torque coefficient, ρ_{ref} the density of air, R_{rotor} the rotor radius and $V_{freeStream}^2$ is the *freeStreamVelocity*.

The power produced is given by:

$$P = T * \omega \qquad \text{Equation 3.72}$$

4 Fish Farm Cases

This chapter will first introduce the companies that have made this thesis possible. Thereafter, the wind measurement campaign done on Tallaksholmen will be presented. The result from the campaign and how the inlet boundary conditions were set will be discussed. Lastly, a second fish farm barge that will be used to show the flexibility of the framework made in this thesis will be introduced.

4.1 Gwind and Grieg Seafood Rogaland

Gwind is a small Norwegian company based in Stavanger which specializes in off-grid energy solutions. This included among others solar- and wind energy, batteries and backup diesel generators. Gwind has earlier developed and tested a 1 kW floating VAWT [48]. In May 2017, Gwind mounted a VAWT on Skrova which will power the lighthouse, see figure Figure 21. The 100-year-old lighthouse is owned by Kystverket [49].

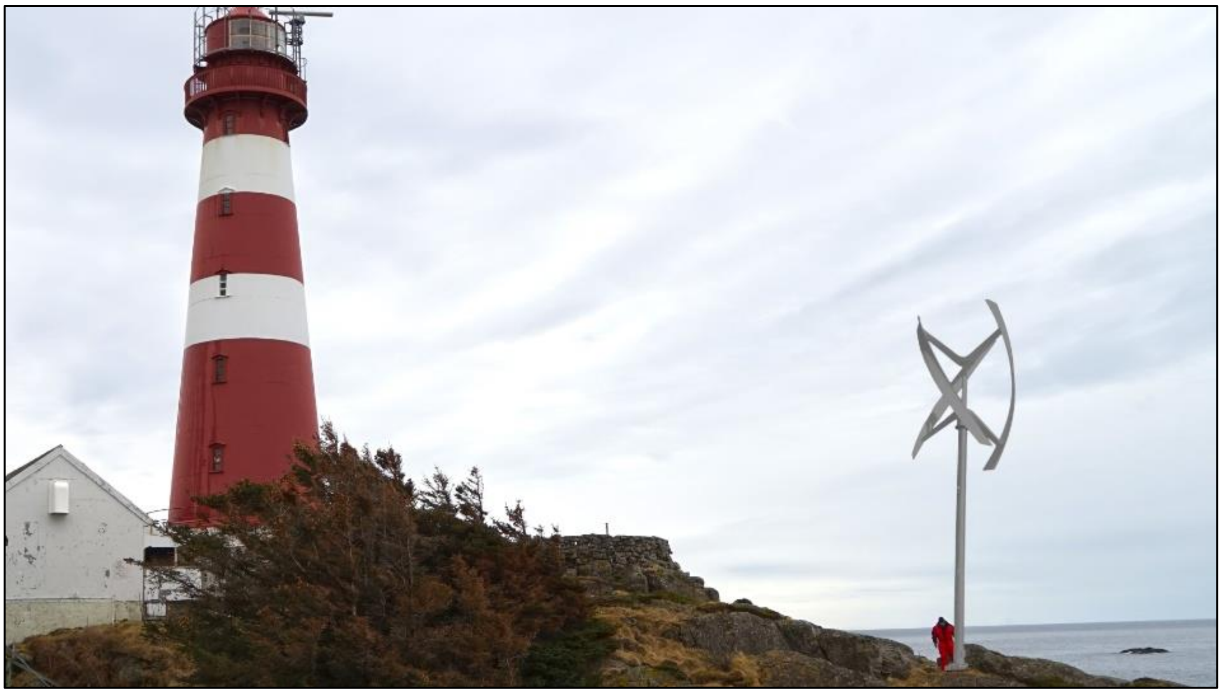


Figure 21. Skrova lighthouse, with VAWT. Illustration: Gwind.

Gwind has been targeting the aquaculture industry and have lately been doing collaborations with Grieg Seafood Rogaland. This gave the author of this thesis the opportunity to visit the fish farms that are being studied in this thesis, together with Gwind. Grieg Seafood ASA is one of the world's leading fish farming companies, specializing in Atlantic salmon. Tallaksholmen, the fish farm visited is a part of Grieg Seafood Rogaland (GSFR). GSFR farms salmon on 20

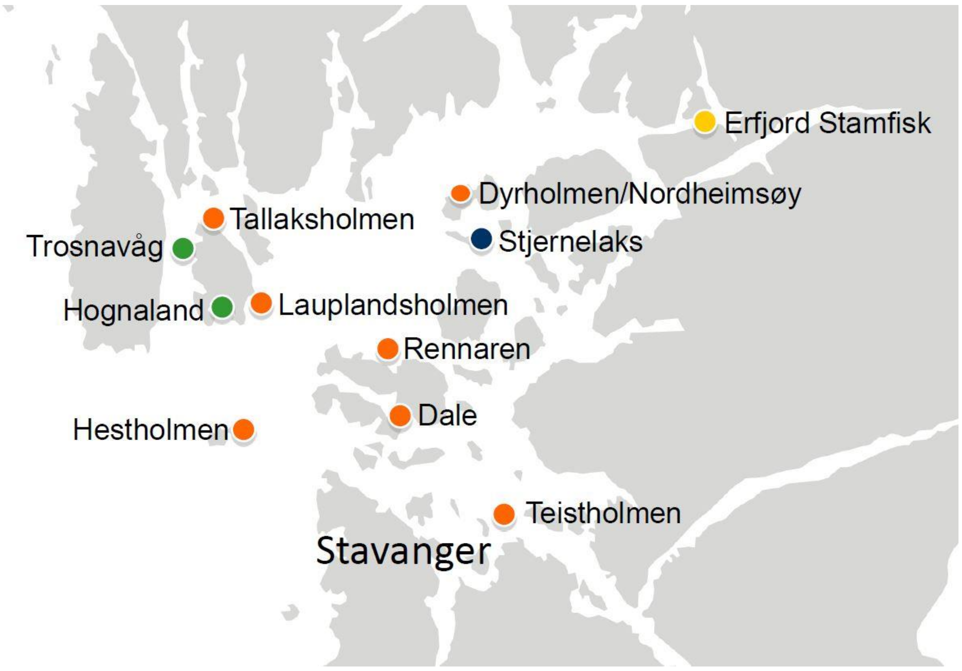


Figure 22. Grieg Seafood Rogaland. The colors indicate; Yellow – Broodstock, Green – Freshwater, Red – Seawater, Blue-Harvest. [51]

grow-out licenses and two smolt licenses in Rogaland [50]. GSFR have fish stations all over Rogaland as seen in Figure 22.

4.2 Tallaksholmen

The first fish farm case is Tallaksholmen, located in Bokn, Rogaland. This fish farm is a seawater fish farm where the fish grows until they are ready to be harvested.

The picture in Figure 23 was taken when leaving Tallaksholmen fish farm feed barge.

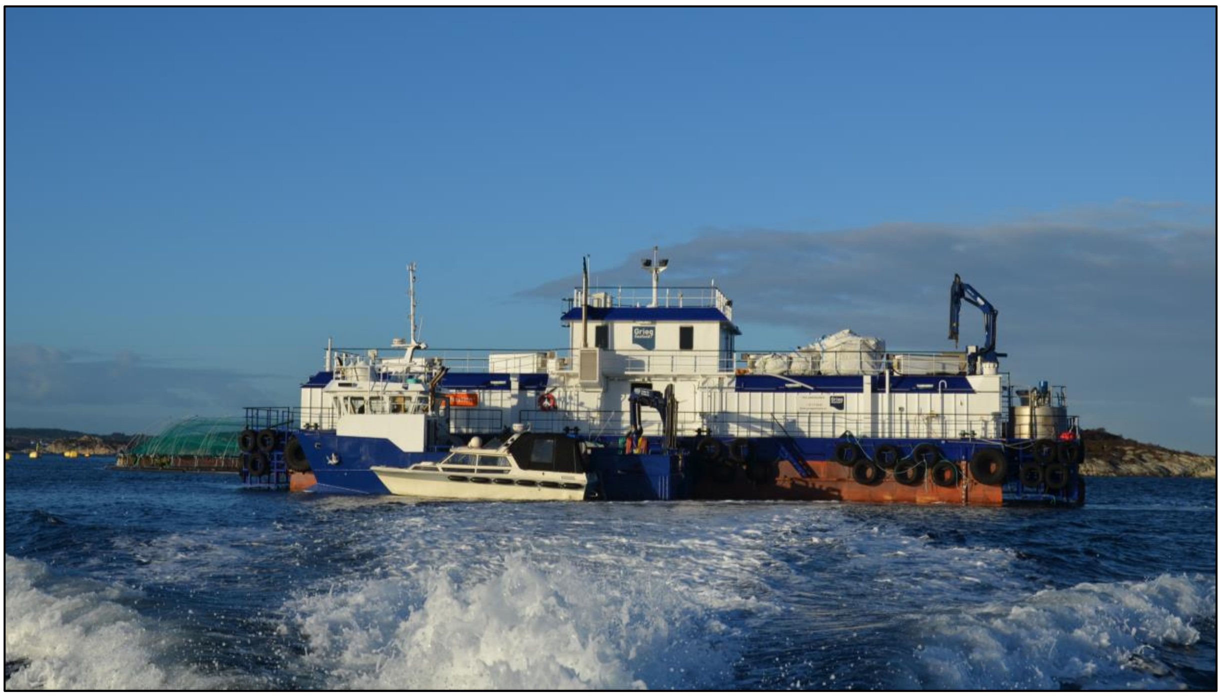


Figure 23. Leaving Tallaksholmen fish farm feed barge

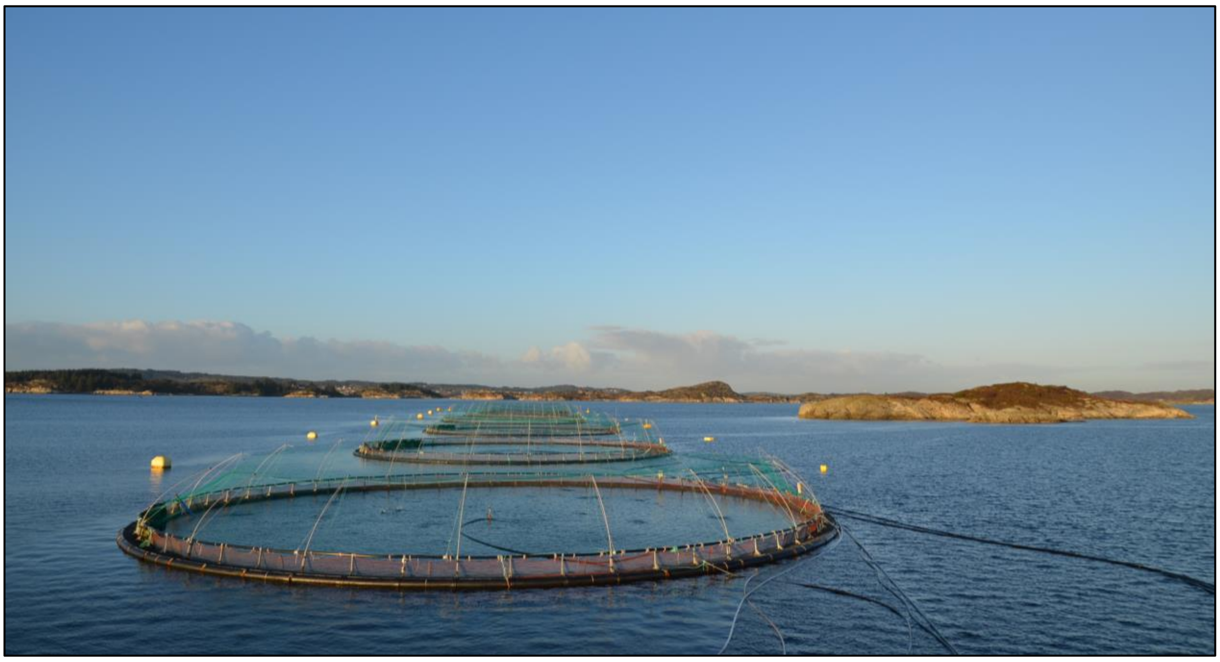


Figure 24. Tallaksholmen, fish farm nets.

Tallaksholmen has a total of six fish farm nets. The fish farm feed barge stores the food in six large silos. The reason for the visit was to dismantle the wind measurement equipment.

4.2.1 Survey

The wind campaign measurement was done in December 2016 over a period of 14 days.

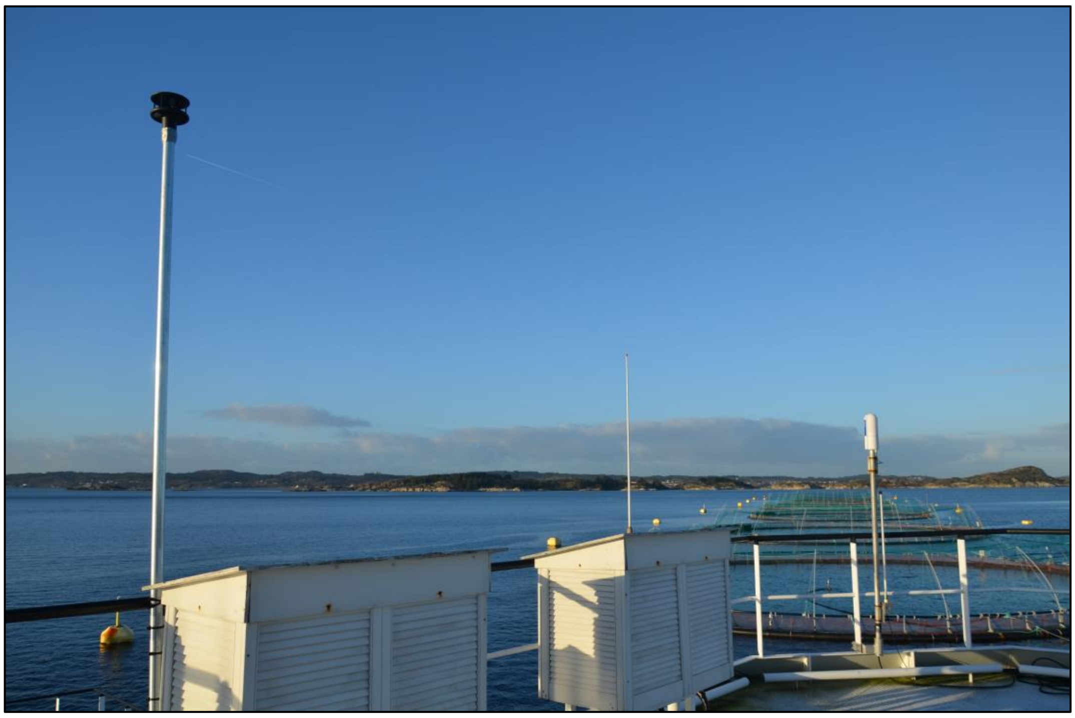


Figure 25. Wind measurement system. The wind sensor is a WindSonic ultrasonic wind sensor.

There was a problem with the power so the survey was done in two periods with the total of 10 days. The WindSonic sampling frequency is $4\ Hz$. The wind velocity and direction has been averaged each hour, then logged. The data is shown graphicly in Figure 26:

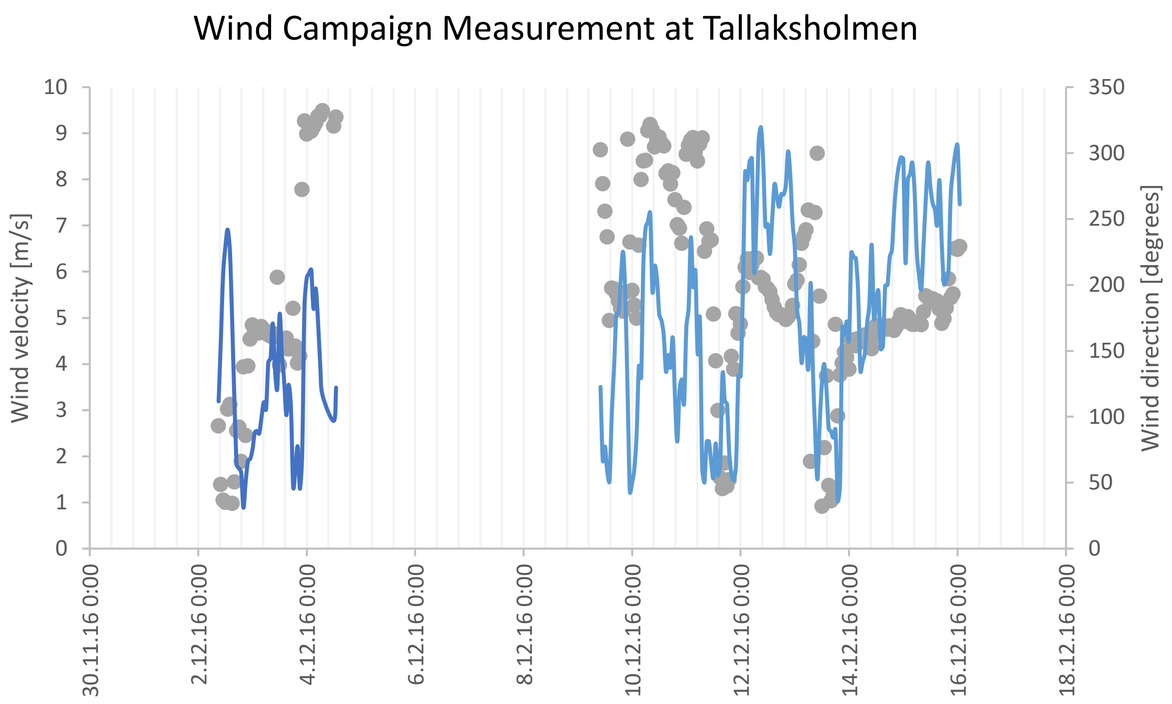


Figure 26. Wind campaign measurement at Tallaksholmen

This data was compared using with Kvitsøy, which is the closest wind measurement station. The data from Kvitsøy was obtained through eklima. A standard regression analysis was conducted to estimate the relationships between the wind speed and direction, on

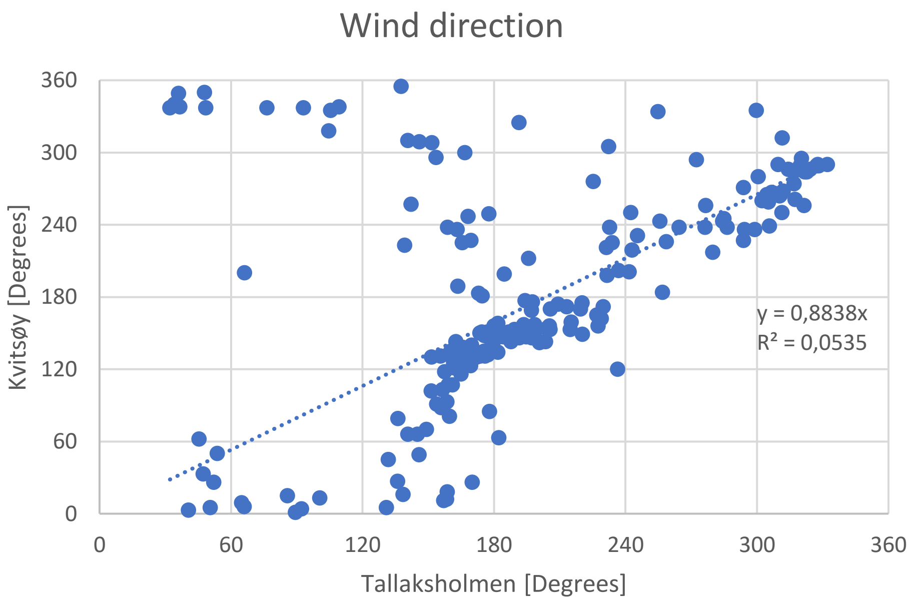


Figure 27. Regression analysis. Wind direction, Kvitsøy and Tallaksholmen.

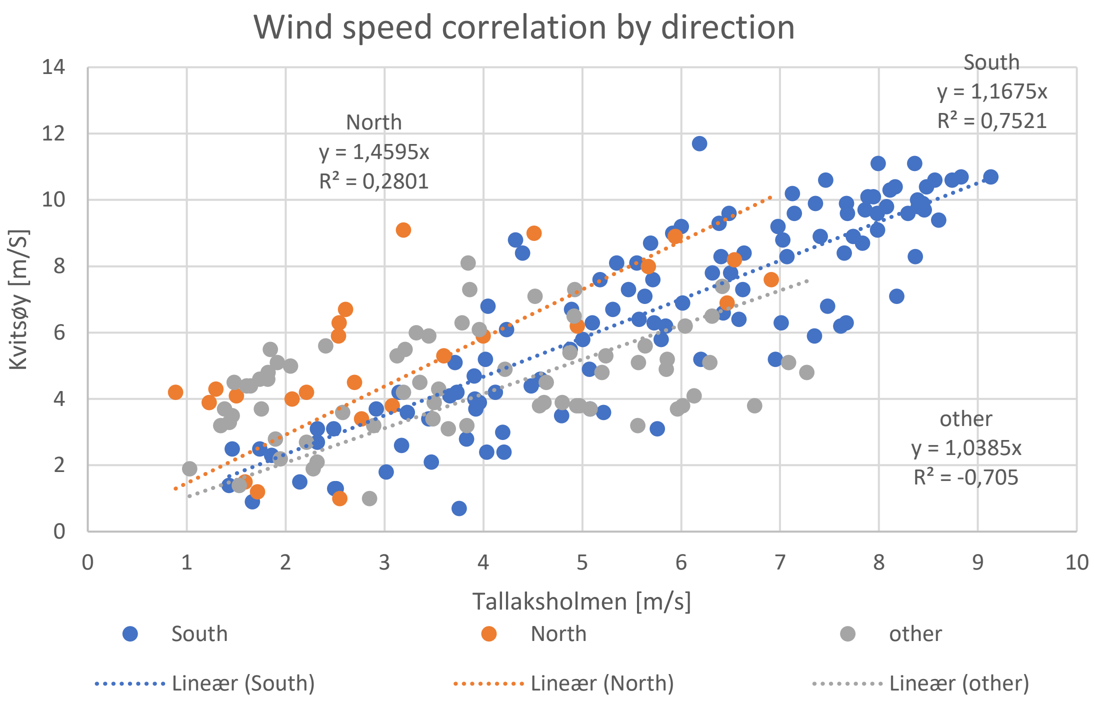


Figure 28. Regression analysis. Wind speed correlation by direction, Kvitsøy and Tallaksholmen.

Tallaksholmen and Kvitsøy. The results wind direction regression analysis is shown in Figure 27, and the wind speed correlation by direction in Figure 28.

From these results, one can see that the wind direction correlates good. The wind speed from South does not correlate very well but North do, this is most likely do the position of the fish farm feed barge. North of the fish farm the flow is not disturbed, whereas South of the fish

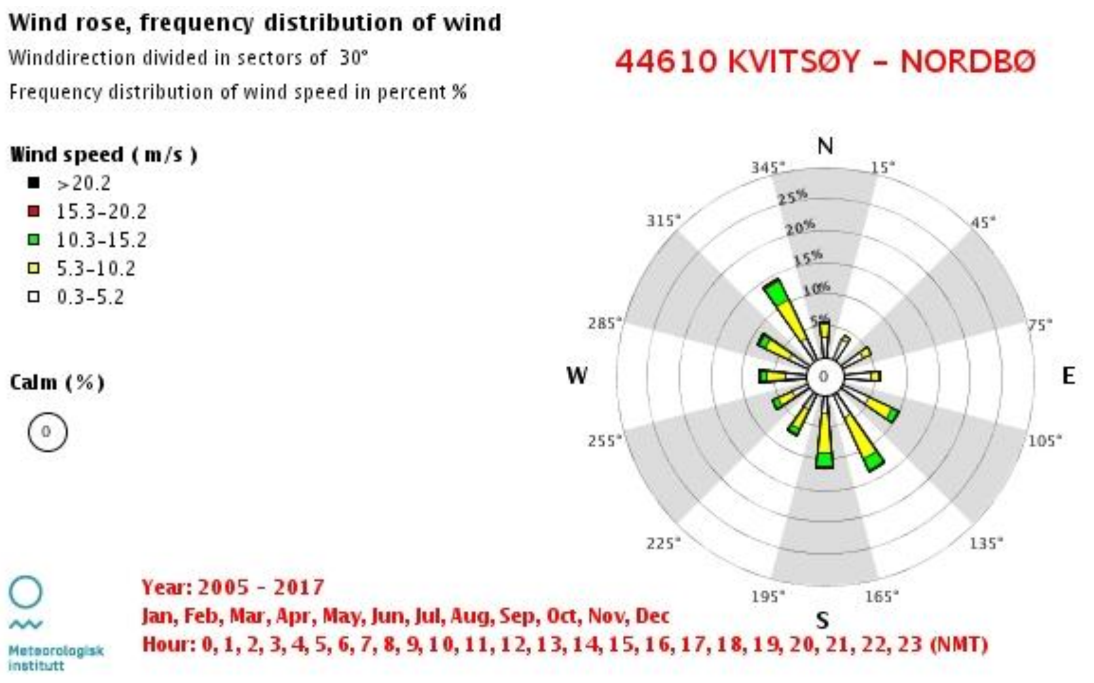


Figure 29. Wind rose from Kvitsøy. Data from 2005-2017. Obtained from eklima.

farm Bokn is obstructing the wind. With this data, the boundary conditions for the inlet has been set with a velocity of $5.7\, m/s$ at $10m$ height.

A wind rose from Kvitsøy is shown in Figure 29. The wind rose has been used in the simulations to correct for direction in the second set of simulations.

One important thing to notice when modeling fish farm barges, is that they are always oriented perpendicular to the main wind direction, which make them easy to implement in the model coordinate system. The barges are either oriented with the wide side against the wind to get the waves to pass in a shorter time, or the narrow up against to distribute the wave force more evenly.

Tallaksholmen is oriented with the wide side against the main wind direction, so the “North” used in the model is oriented $-30°$ from the true North.

4.3 Nordheim

The second case is Nordheim fish farm. The inlet conditions from Tallaksholmen have been used since there was not conducted a wind campaign measurement. Nordheim fish farm barge is oriented with the narrow side in the main wind direction.

This case will be run to show the flexibility of the framework that has been developed.

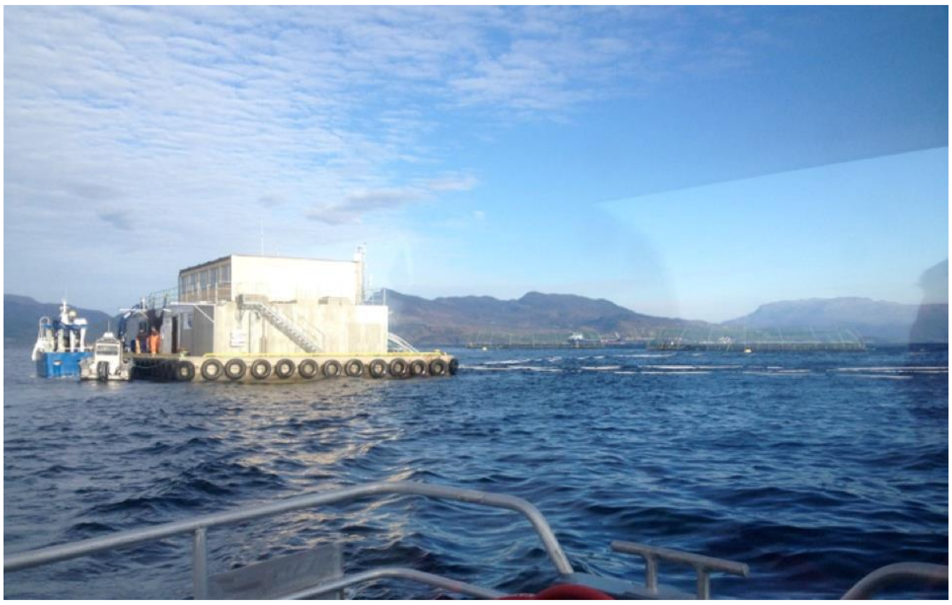


Figure 30. Nordheim fish farm feed barge. Picture provided by Gwind.

5 Modelling approach and experimental setup

This chapter will present how the experimental setup was approached. First geometric modeling of the fish farm barge and the mesh generation of the model is presented. The case setup is also presented, together with the boundary conditions and numerical schemes that are used. Lastly, the implementation of the actuator line model and its setup in fvOptions is presented.

5.1 Geometric modeling

There was no three-dimensional model of the fish farm barge available, which had to be made. GSFR provided a scanned two-dimensional computer aided design (CAD) drawing, in a portable document format (PDF) format, of the barge. Due to the inadequate quality of the scan, Adobes Acrobat Readers DC measurements tool were used to set the different measurements on the drawing, based on some measurements that could be read. The CAD software used for geometric modeling was Autodesk Inventor Professional 2017, with a student license. The model is seen cleaned for simulations in Figure 31:

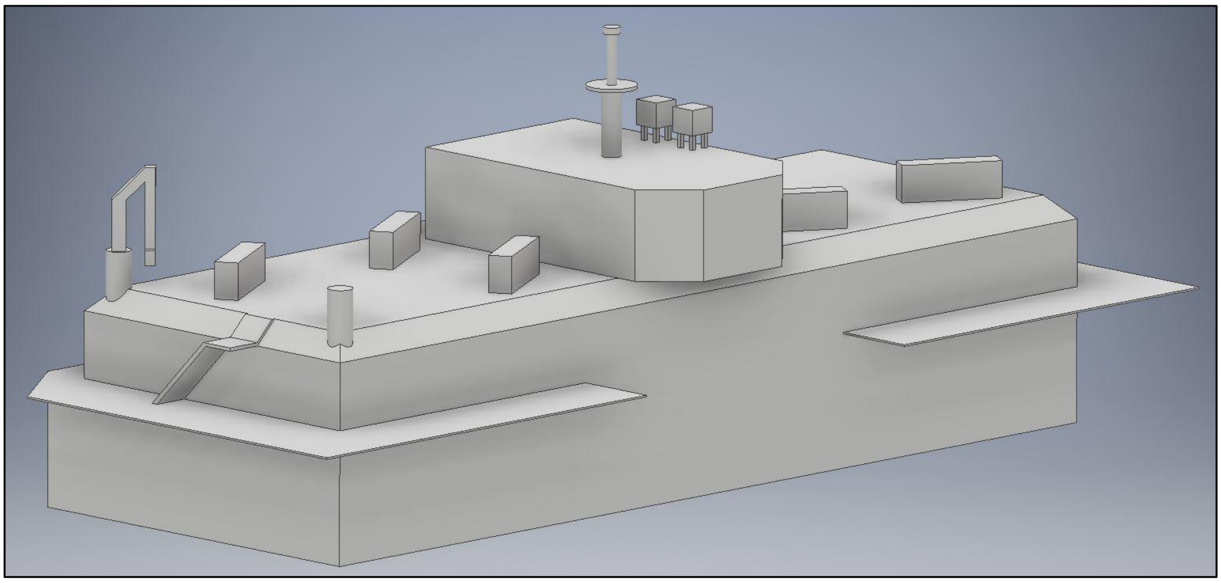


Figure 31. Fish farm feed barge CAD assembly, cleaned for simulations.

GSFR decided that if there was to be placed a wind turbine on the barge, it would be mounded at a specific place, see Figure 32. Figure 32 also displaces the main dimensions of the feed barge. Inventor was used for exporting the geometry to an ASCII stereolithography file (STL). The STL file has been dubbed Foater_STL.

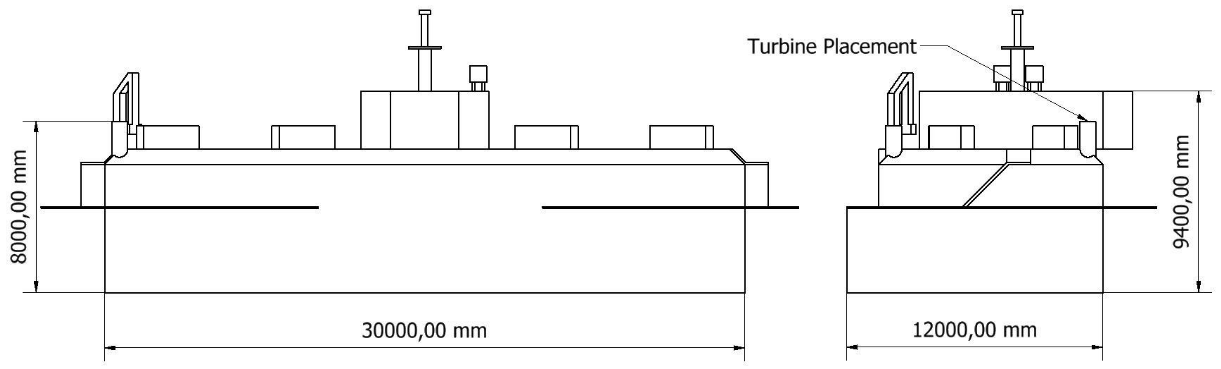


Figure 32. Fish farm feed barge drawing, with dimensions and turbine placement.

5.2 Mesh generation

The mesh generation of this project is done in two main steps, where first a background mesh is created with *blockMesh* utility and then the *snappyHexMesh* utility is used with different controls activated to, import the fish farm barge geometry and refining around the ALM.

The background mesh was generated using the *blockMesh* utility already present in OpenFOAM. The *blockMesh* utility creates blocks with hexahedral cells. This is done by defining vertices with coordinates in the dictionary file *blockMeshDict*, which is located in *system*. The number of cells in each direction is set and last the boundary patches is defined. The number of cells have been chosen based a mesh dependence test, where relevant parameters such as velocity and turbulent kinetic energy has been observed. This will be further presented in section 6.1. The dimensions for the base mesh is $100m\ x\ 100m\ x\ 20m$ in x-, y- and z direction respectively. This resulted in a number of cells in each direction of 80, 80 and 16 in x-, y- and z direction respectively. This is done so that the cells are background mesh consists of hexahedral cells with the aspect ratio of one. This results in a background mesh of $102.4e3$ cells with a cell length of $1.25m$.

snappyHexMesh is used to finalize the mesh generation. *snappyHexMesh* was used with castellating- and snapping controls activated. First, the surface features must be extracted from the STL file, this is done by running *surfaceFeatureExtract* utility. This creates a set of files which contains information about the distinctive features of the STL file, level refinement of the features is set at one, which will split the cells in two.

The castellating will approximate the tri-surface from the STL file, by splitting or removing cells that are inside in this case of the STL, because in this case the mesh is generated outside of the tri-surface. The cells that have been split will refine the mesh near the surface, the level is set to a minimum and maximum of three, which will split the cells 2^3 times. This results in a cell length of $0.15626m$ around the tri-surface. There are also refinement regions around the ALM, which will be discussed in 0. At last the castellating will remove the cells which in this case is inside the tri-surface. The snap controls will modify the remaining mesh to reconstruct the surfaces of the STL file. The background mesh with the turbine and fish farm feed barge is shown in Figure 33 and only the feed barge in Figure 34:

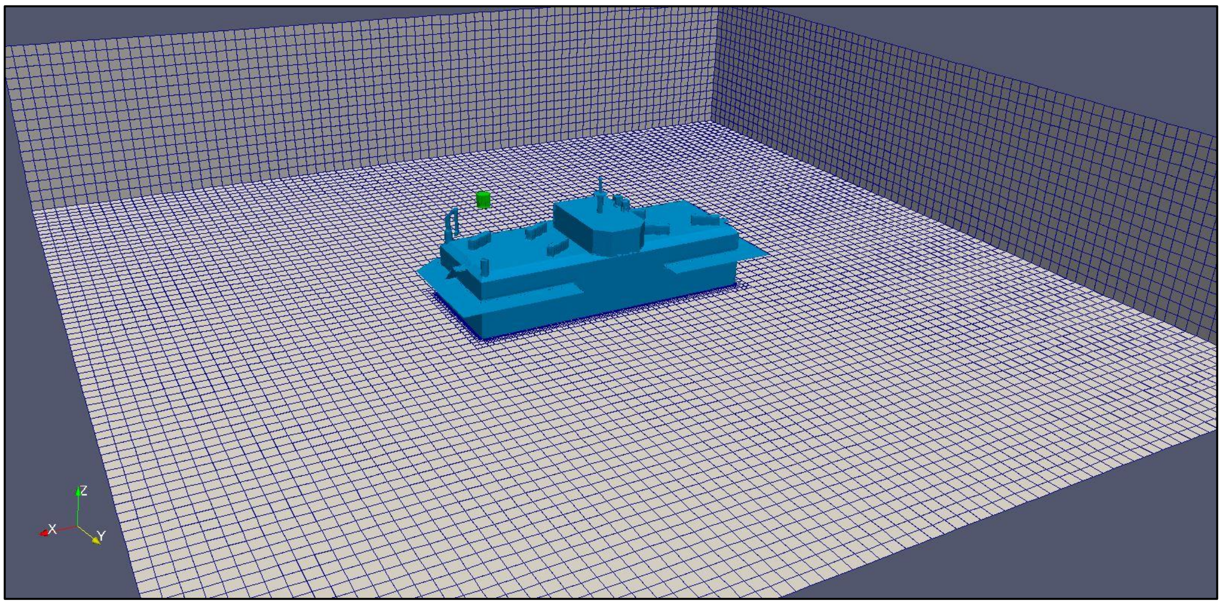


Figure 33. Background mesh with turbine (green) and fish farm feed barge (blue)

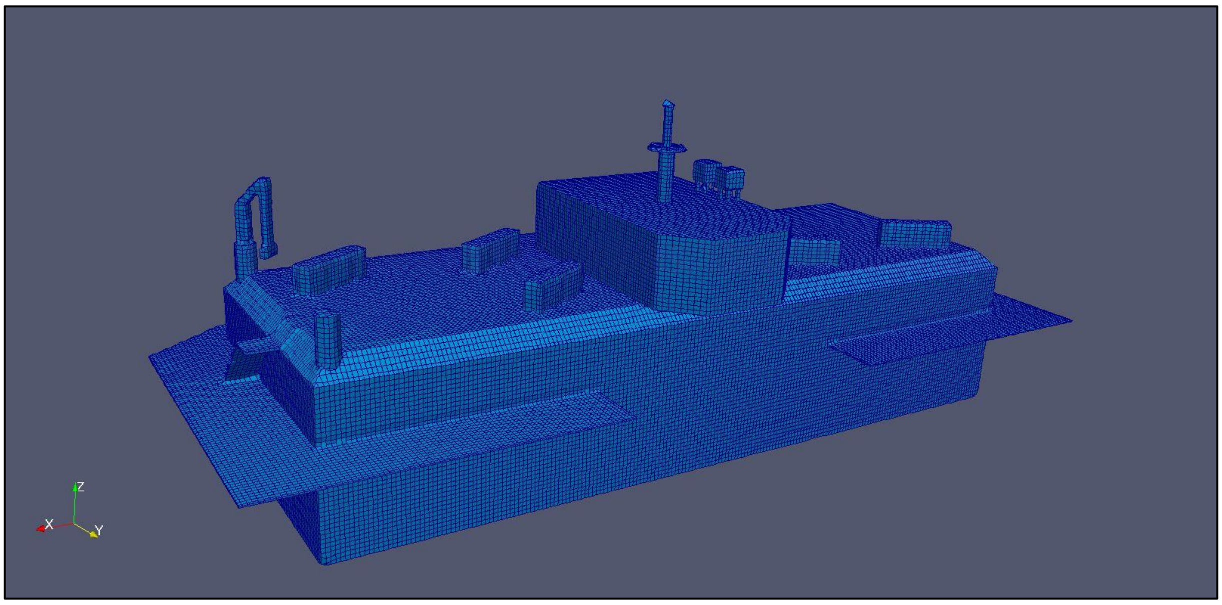


Figure 34. Fish farm feed barge, surface with edges

5.2.1 ALM implementation

The actuator line model, referred to as the turbine, is implemented in case by defining the position of the turbine in the *fvOptions*. The turbine must be defined as a zone, which is done by the *topoSet* utility. The focus in this section is the meshing of the turbine.

The mindset when defining the refinements around the turbine, was that it should be identical for every direction. So that the inlet/outlet could be changed without having to change the refinements around turbine. The refinement zones are cuboids where the length and width is equal. The refinement was chosen to fit the number of cells in the turbine cell zone of the reference model [5].

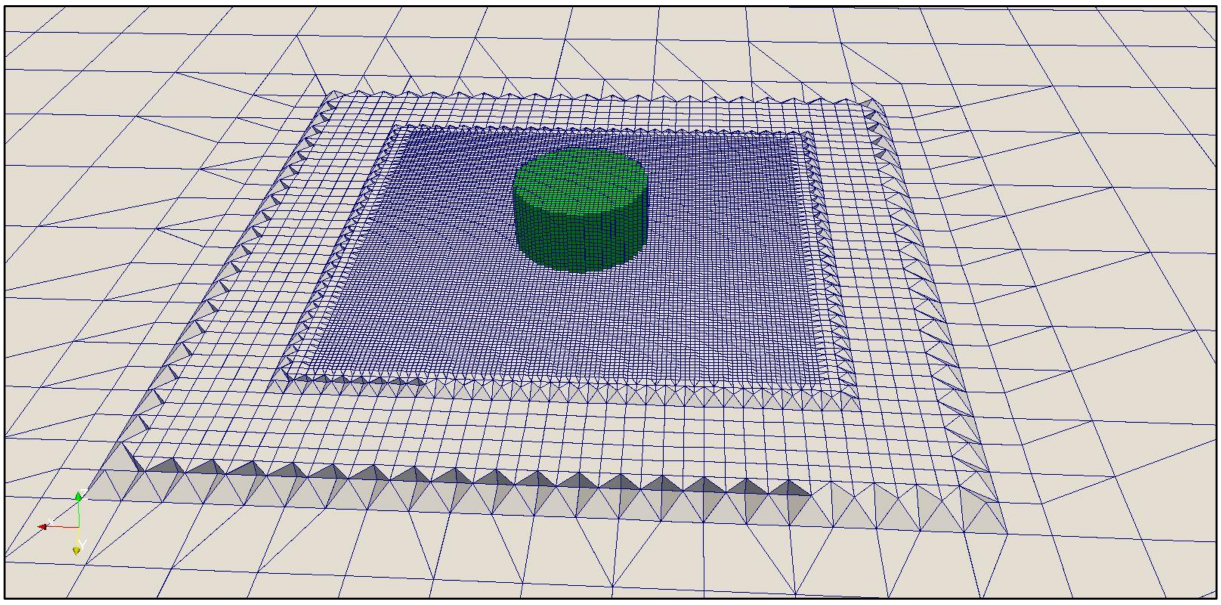


Figure 35. Refinement boxes around turbine, seen from top

There are two refinements boxes, this is done by defining two *searchableBox*. The outer refinement box is $6.25m\ x\ 6.25m\ x\ 3.75m$, with level 3 refinement, this results in cell length of $0.15626m$. The inner refinement box is $4.25m\ x\ 4.25m\ x\ 1.6m$, with level 5 refinement, which results in a cell length of $0.039m$, see Figure 35. The reference model has a cell length of $0.038m$ within the turbine. The boxes are centered at the turbines center. To make this process less tedious an Excel worksheet has been made, here one can input the coordinates for the turbine and the various coordinates that must be changed are outputted.

The refinement around the turbine is seen in from the front in Figure 36 and from the side in Figure 37:

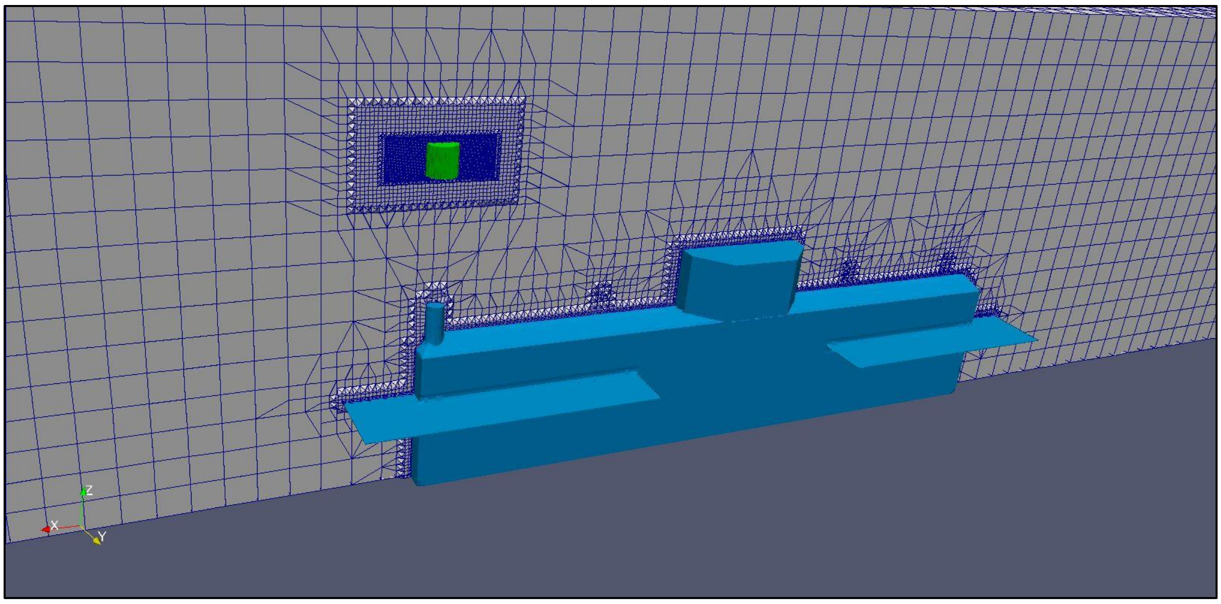


Figure 36. Mesh seen from front

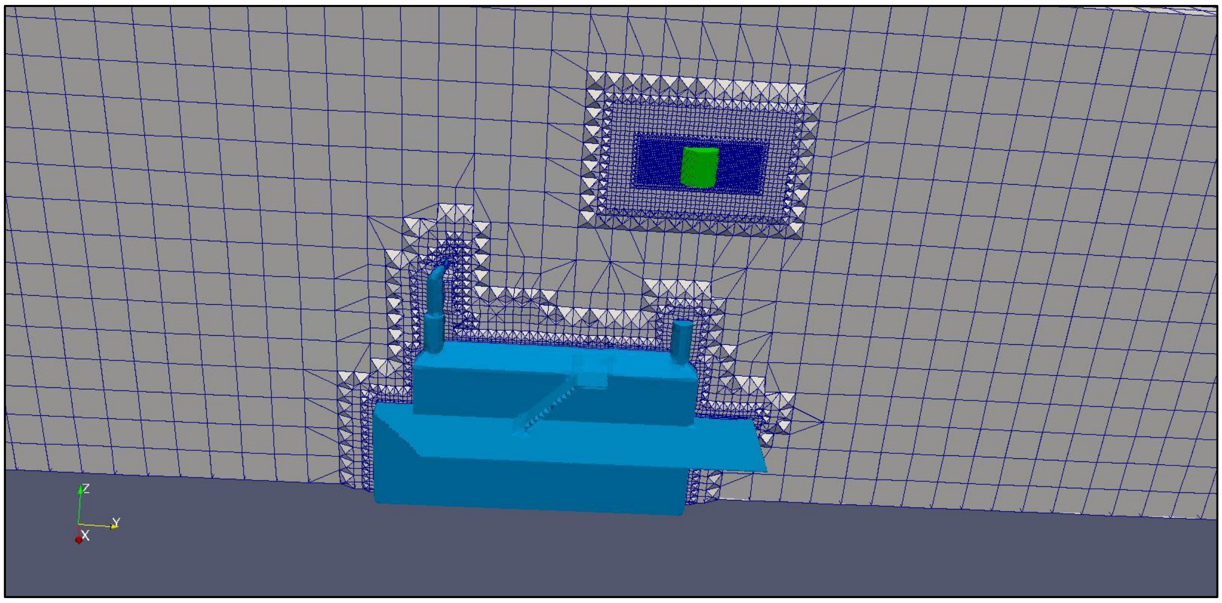


Figure 37. Mesh seen from side

As mentioned the turbine must be set as a *cellZone*. This is done by the *topoSet* utility, first a *cellSet* is created with a radius of $0.625m$ and height $1.25m$, with the center in the turbine. Then the *topoSet* converts the *cellSet* to a *cellZone*, which is dubbed turbine. The *cellZone* consist of 25984 cells, see logfile at 9.3.2.

The *cellZone* is shown in Figure 38 both cut and full turbine.

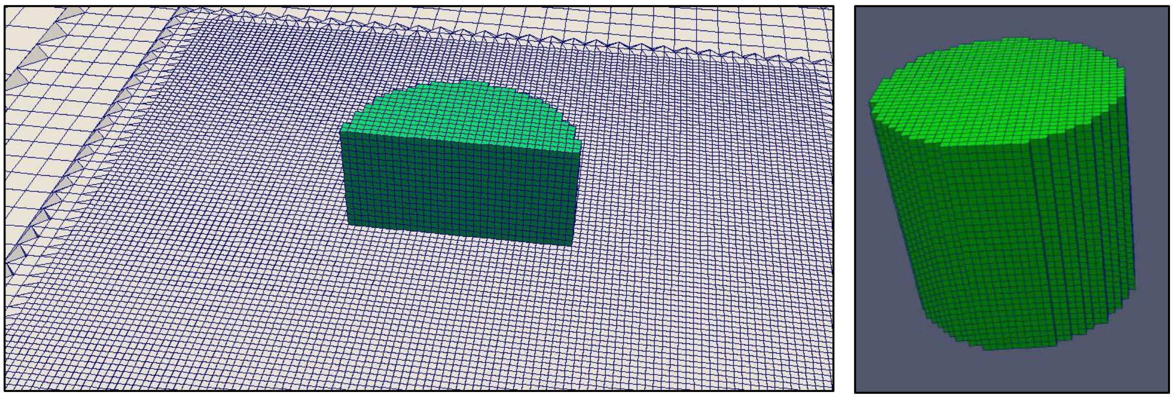


Figure 38. cellZone cut and full cellZone, respectively

The finalized mesh resulted in just above $700k$ cells, as seen in the log file for *checkMesh* in 9.3.1. The final mesh is shown in Figure 39:

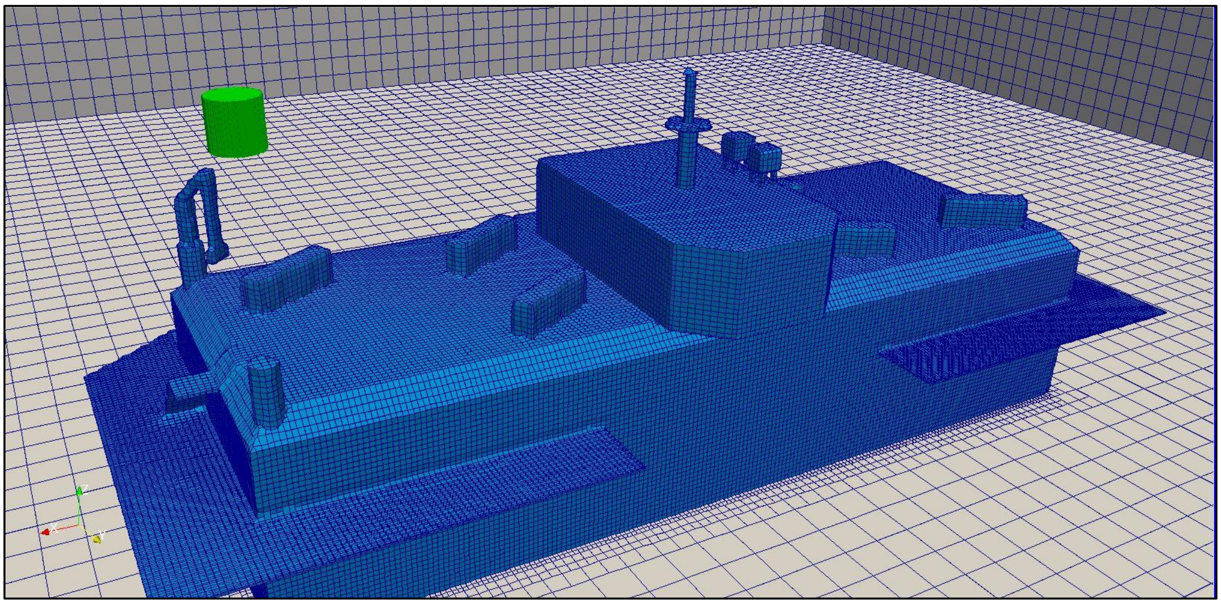


Figure 39. Final mesh, with turbine (green) and fish farm feed barge (blue)

5.3 Case setup

This chapter includes a description of how the case is set up. Boundary conditions, numerical schemes, solvers and algorithm control, time control and ALM setup is discussed. At last instructions on how to move the turbine and what need changing in doing so is given.

5.3.1 Boundary conditions

The boundary conditions, for the $k - \epsilon$ turbulence model, is given in Table 2 and Table 3. The inlets are defined by OpenFOAM class function *atmBoundaryLayer* which is described in section 3.2.8, the parameters are set in the *ABLConditions* file:

```
Uref                    5.7;
Zref                    10;
zDir                    (0 0 1);
flowDir                 (0 -1 0);
z0                      uniform 0.01;
zGround                 uniform 0.0;
value                   $internalField;
```

Additional information is needed for the ABL to function, more accurate the turbulent kinetic energy and dissipation rates at the inlet [43], is defined in *internalField*:

```
flowVelocity            (0 -5.7 0);
pressure                0;
turbulentKE             0.0364;
turbulentEpsilon        4.46e-3;       // Guess ~ 0.09*k**(3/2)/l
```

The ABL has been tested and will be presented in section 6.1.

Table 2. Boundary conditions for velocity, epsilon and k

Patch	Boundary conditions		
	U	*epsilon*	*k*
`internalField`	`uniform $flowVelocity;`	`uniform $turbulentEpsilon;`	`uniform $turbulentKE;`
`dimensions`	`[0 1 -1 0 0 0 0];`	`[0 2 -3 0 0 0 0];`	`[0 2 -2 0 0 0 0];`

`inlet`	`type atmBoundaryLayerInletVelocity;`
	`#include "include/ABLConditions"`
		`type atmBoundaryLayerInletEpsilon;`
		`#include "include/ABLConditions"`
			`type atmBoundaryLayerInletK;`
			`#include "include/ABLConditions"`

`outlet`	`type inletOutlet;`
	`inletValue uniform (0 0 0);`
	`value $internalField;`
		`type inletOutlet;`
		`inletValue uniform $turbulentEpsilon;`
		`value $internalField;`
			`type inletOutlet;`
			`inletValue uniform $turbulentKE;`
			`value $internalField;`

`Floater_STL`	`type uniformFixedValue;`
	`uniformValue (0 0 0);`
	`value uniform (0 0 0);`
		`type epsilonWallFunction;`
		`Cmu 0.09;`
		`kappa 0.4;`
		`E 9.8;`
		`value $internalField;`
			`type kqRWallFunction;`
			`value uniform 0.0;`

`ground`	`type uniformFixedValue;`
	`uniformValue (0 0 0);`
	`value uniform (0 0 0);`
		`type epsilonWallFunction;`
		`Cmu 0.09;`
		`kappa 0.4;`
		`E 9.8;`
		`value $internalField;`
			`type kqRWallFunction;`
			`value uniform 0.0;`

`side (wall-s, n, e, w)`	`type slip;`	`type slip;`	`type slip;`
`top`	`type zeroGradient;`	`type zeroGradient;`	`type zeroGradient;`

Table 3. Boundary conditions for nut and pressure

Patch	Boundary conditions	
	nut	*p*
`internalField`	`uniform 0;`	`uniform $pressure;`
`dimensions`	`[0 2 -1 0 0 0 0];`	`[0 2 -2 0 0 0 0];`

`Inlet`	`type calculated;`
	`value uniform 0;`
		`type zeroGradient;`

`Outlet`	`type calculated;`
	`value uniform 0;`
		`type uniformFixedValue;`
		`uniformValue constant $pressure;`

`Floater_STL`	`type nutUSpaldingWallFunction;`
	`value uniform 0.0;`
		`type zeroGradient;`

`Ground`	`type nutkAtmRoughWallFunction;`
	`z0 $z0;`
	`value uniform 0.0;`
		`type zeroGradient;`

`side (wall- s, n, e, w)`	`type slip;`	`type slip;`
`Top`	`type zeroGradient;`	`type zeroGradient;`

5.3.2 Numerical Schemes

The numerical schemes that is used for discretization is set in the *fvSchemes* file, the selected schemes for this project is shown in Table 4.

Table 4. Selected numerical discretization schemes in fvSchemes dictionary

Groups	Numerical Scheme
`ddtSchemes`	`default          backward;`

`gradSchemes`	`default          Gauss linear;`
	`grad(p)          Gauss linear;`
	`grad(U)          Gauss linear;`

`divSchemes`	`default           none;`
	`div(phi,U)        Gauss linearUpwind grad(U);`
	`div(phi,k)        Gauss upwind;`
	`div(phi,epsilon)  Gauss upwind;`
	`div(phi,R)        Gauss upwind;`
	`div(R)            Gauss linear;`
	`div(phi,nuTilda)  Gauss upwind;`
	`div((nuEff*dev(T(grad(U)))))   Gauss linear;`
	`div((nuEff*dev2(T(grad(U))))) Gauss linear;`

`laplacianSchemes`	`default          Gauss linear corrected;`
`interpolationSchemes`	`default          linear;`
`snGradSchemes`	`default          corrected;`

`fluxRequired`	`default          no;`
	`p;`

Detailed description of the discretization schemes will not be done in this thesis, but references are made to [34, pp. 369-411, 52].

5.3.3 Solver and algorithm control

The solver and algorithm control is set in the *fvSolution* dictionary. In this thesis, *pimpleFoam* is used, which is a large time-step transient solver for incompressible, turbulent flow, using the PIMPLE (merged PISO-SIMPLE) algorithm [53]. The PISO and SIMPLE algorithms was presented in 3.2.6.

The pressure solver used is a generalized method of geometric-algebraic multi-grid (GAMG), which is used for pressure equation on large grids [34, pp. 356]. The solver preforms calculations a coarse grid, then map this on a finer grid as initial guess for the calculation to obtain a more accurate solution on the fine mesh. The number of refinements can be set. The solver is being smoothed with *DICGaussSeidel*, which is a diagonal incomplete-Cholesky with Gauss-Seidel, this smoother can offer better convergence for bad matrices than *GaussSeidel* [37, pp. U-82]. The rest of the parameters are solved with the *smoothSolver* with *symGaussSeidels* smoother. The solvers selected is shown in Table 5.

Table 5. Solvers selected in fvSolution dictionary

Equation	*pimpleFoam*

`Pcorr`	`solver GAMG;`
	`tolerance 1e-4;`
	`relTol 0;`
	`smoother DICGaussSeidel;`
	`cacheAgglomeration no;`
	`nCellsInCoarsestLevel 10;`
	`agglomerator faceAreaPair;`
	`mergeLevels 1;`
	`maxIter 50;`

`P`	`$pcorr;`
	`tolerance 1e-6;`
	`relTol 0.01;`

`pFinal`	`$p;`
	`tolerance 1e-6;`
	`relTol 0;`

`"(U|k|epsilon|nuTilda|R)"`	`solver smoothSolver;`
	`smoother symGaussSeidel;`
	`tolerance 1e-6;`
	`relTol 0.1;`

`"(U|k|epsilon|nuTilda|R)Final"`	`$U;`
	`relTol 0;`

Table 6. Settings for solution algorithm control

PIMPLE

`nOuterCorrectors 1;`
`nCorrectors 2;`
`nNonOrthogonalCorrectors 0;`
`pRefCell 1001;`
`pRefValue 0;`

Table 6 shows the settings for solution algorithm control, here the *nOuterCorrectors* are how many loops is done on each timestep. When this is kept as one the solver replicates the PISO algorithm. The *nCorrectors* specifies how many correctors in the PISO is done. Then the *nNonOrthogonalCorrectors* should correspond to the mesh for the case being solved, i.e. 0 for an orthogonal mesh and up to 20 for most non-orthogonal meshes [34, pp. 241-51, 37, pp. U-84]. The pressure reference stores the relative value of pressure and the position, and is need for the solver to run [37, pp. U-84].

5.3.4 Time control

The time settings are set in the *controlDict* dictionary. Both the *endTime* and *deltaT* has been considered more closely in this thesis. The *endTime* is mostly decided from when the ABL stabilizes, this study will be presented in section 6.1. The other parameter that has been studied is the *deltaT*, here one must take into consideration; the rotational speed of the turbine and the cell length inside the turbine *cellZone.* The timestep study is presented in section 6.2.2.

5.3.5 Actuator Line Model setup

The ALM is defined in the *fvOptions* dictionary, see 9.4.2. The turbinesFoam library have the option of choosing either a vertical- or horizontal axis turbine, in this project the vertical axis, dubbed *crossFlowTurbineAL*, is being used.

The main parts defining the turbine coefficients:

crossFlowTurbineALSourceCoeffs	

First part	`fieldNames          (U);`
	`selectionMode       cellSet;`
	`cellSet             turbine;`
	`origin              (14.3 5.3 14);  // Center wind turbine`
	`axis                (0 0 1);        // Negative for opposite rotation`
	`rotorRadius         0.5;`
	`freeStreamVelocity  (0 -5.7 0);`
	`tipSpeedRatio       1.9;`
	`tsrAmplitude        0.0;            // Amplitude for TSR oscillation`
	`tsrPhase            0.0;            // Angle of first peak (rad)`

ALM parameters	`addedMass           // Unsteady effect`
	`dynamicStall        // Unsteady effect`
	`flowCurvature       // Flow curvature corrections`

Physical properties	`blades`
	`struts`
	`shaft`
	`profileData`

In the first part of *fvOptions*, the *cellZone* created with the *topoSet* utility earlier is defined as the turbine. Next the center of the wind turbine, the rotation direction and the rotor radius must be set, the center of the turbine which has been set represents a tower height of $6m$, which has been used as "default" height for most of the simulations. Then the *freeStreamVelocity* is set, as discussed in section 3.3, this is used for setting the direction of the velocity vector correct when the solver is calculating and nondimensionalization to get the performance of the turbine. Then *tipSpeedRatio* (TSR) and options for the TSR is set, the TSR is constant.

There is the option of turning on/off unsteady effects such as added mass and dynamic stall. There is also a flow curvature corrector, all these settings have been kept on with the default values.

Last there is the physical properties of the turbine and their profile data. The blades are defined as such:

```
writePerf    true;
nElements    7;
endEffects   on;
elementProfiles (NACA0020);
elementData
( // axialDistance, radius, azimuth, chord, chordMount, pitch
   (-0.5 0.5 0.0 0.14 0.5 0.0)
   ( 0.5 0.5 0.0 0.14 0.5 0.0)
```

When the *writePref* is turned on the solver outputs a file for the actuator line, which writes information each timestep about; location of the blade, attack angle, relative velocity, C_l, C_d and C_m. The number of ALM blade elements is depended on the Gaussian width as discussed in section 3.3.1, this has been more closely tested in 6.2.2.

The struts are defined the same way as the blades, they also use the NACA0020 element profiles. The shaft is defined as a cylinder. The turbine does not consist only of one blade and strut, in this case three blades and struts has been used. The way to add additional blades and struts is to copy the definition of the blade and strut then add an angle offset, in this case each $120°$.

The element profiles are set, which is static input foil coefficients that contains values for different Reynolds numbers. In this case the dataset contains polar values from a NACA0021 airfoil, but the model uses NACA0020, there is only a small difference in thickness so the difference is negligible [5, pp. 4]. The profile data is setup as:

```
tableType    multiRe; // singleRe || multiRe
Re           1.6e5;
data         (#include "NACA0021_1.6e5");
ReList       (4e4 8e4 1.6e5 3.6e5 7e5 1e6 2e6 5e6);
clData       (#include "NACA0021_multiRe_cl");
cdData       (#include "NACA0021_multiRe_cd");
cmData       (#include "NACA0021_multiRe_cm");
```

With the *turbinesFoam* library there are a set of data, the one used in this case contain information for different Reynolds Numbers, then one must set the most accurate number from the *ReList* which in this case is $1.6e5$.

6 Results and discussion

In this chapter, the most important results from the simulations is presented. First the horizontally homogenous turbulent surface layer (HHTSL) analyses are presented. A mesh dependences- and timestep analysis was conducted and the results are presented.

The main significance of this thesis was to build a framework for wind flow simulations over a fish farm feed barge, this will be shown by the simulation result from Tallaksholmen. Additional simulations have been conducted for an additional fish farm feed barge to show the flexibility of the framework. To emphasize the need for CFD simulations, a comparison has been conducted with empirical versus CFD simulation of the wind and power output.

Lastly, results from a cost analysis is presented, this shows how the model could potentially be used for practical engineering problems.

6.1 Mesh dependency study

To ensure that the solution was approximately equal regardless of an increasing grid resolution the HHTSL has been tested. First to find the resolution that was needed, velocity and turbulent kinetic energy (TKE) was observed in the center of the grid, the total grid is $100m \ x \ 100m \ x \ 20m$, with only the block mesh at different cell lengths. The cell length was equal in each direction.

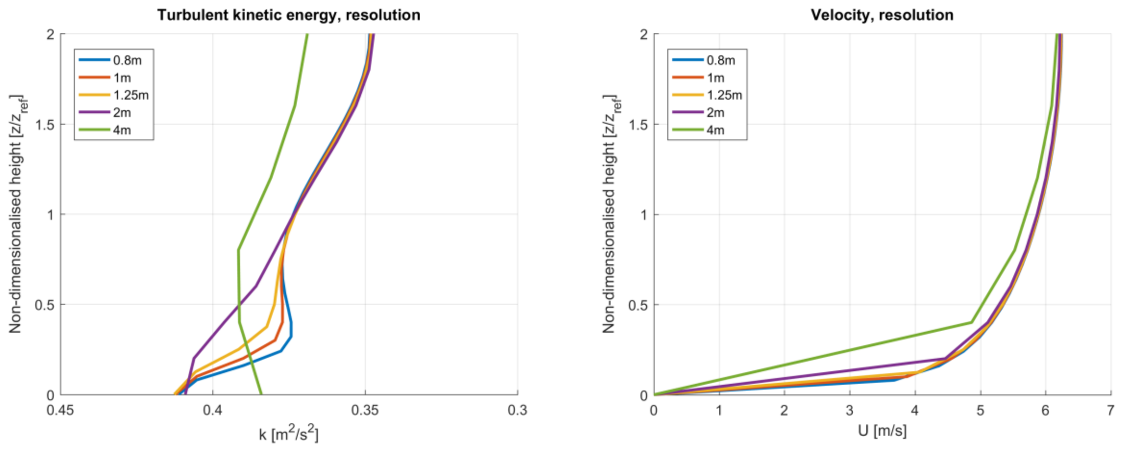


Figure 40. Turbulent kinetic energy and velocity, respectively. At different resolutions of cell lengths. $z_{ref} = 10m$. *The plots shown, is from the timestep at 50 seconds.*

This test showed that when increasing the resolution so that the cell length in x, y and z was $1.25m$, the velocity and TKE profiles was approximately equal to the higher resolution. One thing to keep in mind is that the number of cells increase drastically when the cell length is decreased, the first simulation of a cell length $4m$ and the last of $0.8m$ the number of cells are

3125 versus almost $400k$ before refining around the ALM or geometry. To be able to have a good model that captures the physics of the case, a not to course mesh has been chosen.

The next step was to find at what time the simulation starts to be uniform throughout the mesh. This was done to set a proper runtime for the simulations, the simulation should be stable before the data from the ALM is extracted. The results are shown in Figure 41 and Figure 43.

The TKE profiles is shown in Figure 41:

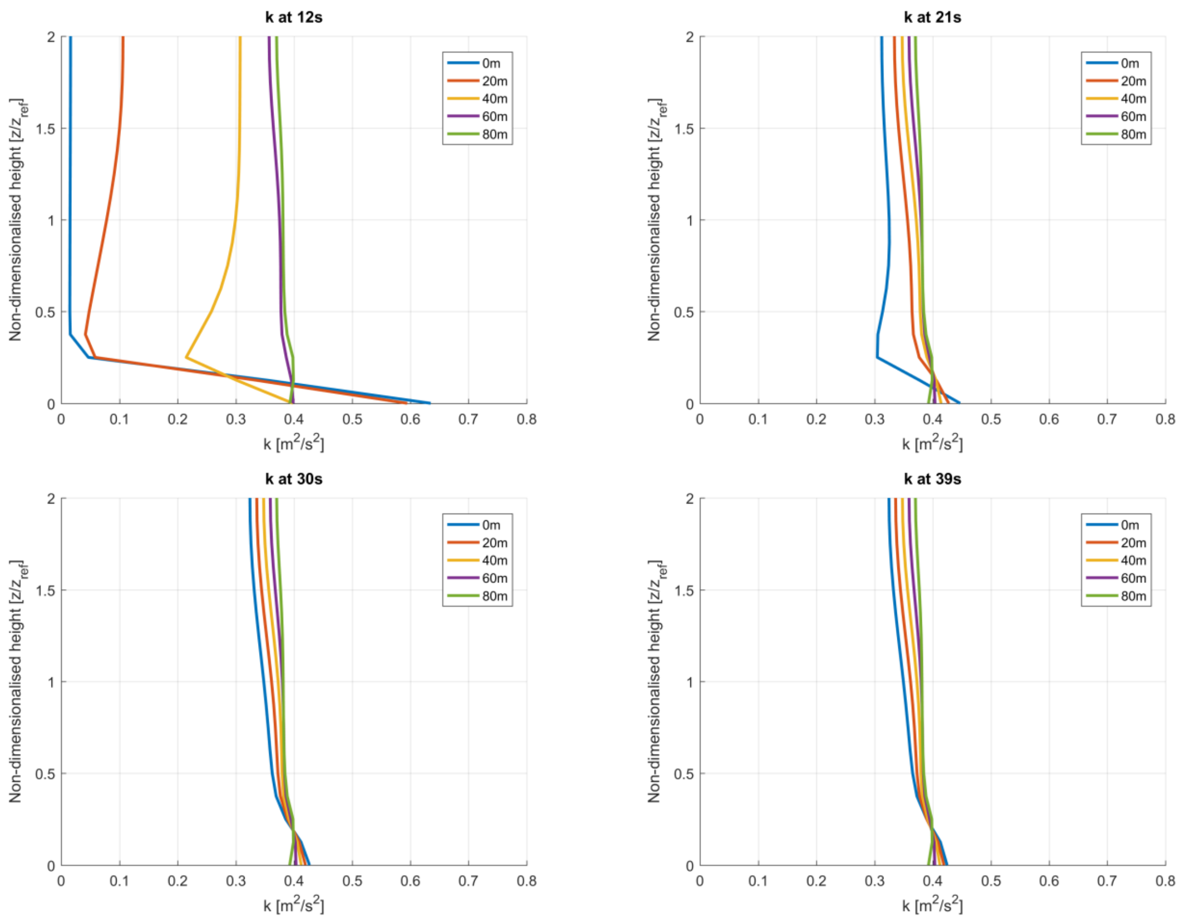


Figure 41. HHTSL k profile evaluated at different timesteps, sampled throughout the mesh

Here one can see that the TKE profile is stable after 30s.

Since the mesh was not that high, a comparison with $z/z_{ref} = 5$ will be done, this was the height used in [43]. This will show how the TKE profiles develops higher in the mesh, the results are shown in Figure 42:

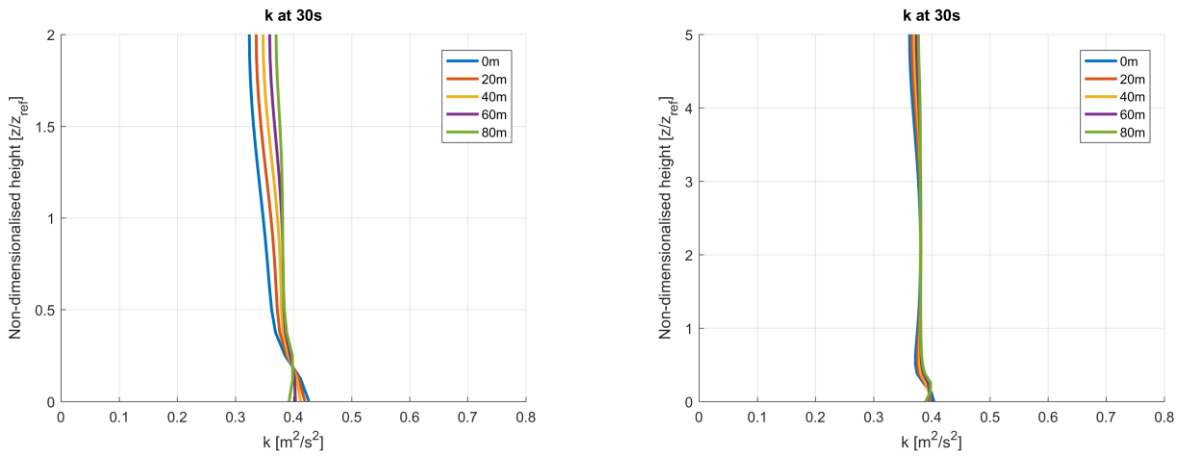


Figure 42. HHTSL k profile evaluated with different mesh sizes: $z/z_{ref} = 2 \text{ and } 5$*, respectively.*

Here one can see that the TKE is developed throughout the mesh. The TKE is more uniform on the higher mesh, this would be preferable, but it has been decided that it's not worth the computational cost due to the more than doubling of cells.

The velocity profiles are shown in Figure 43.

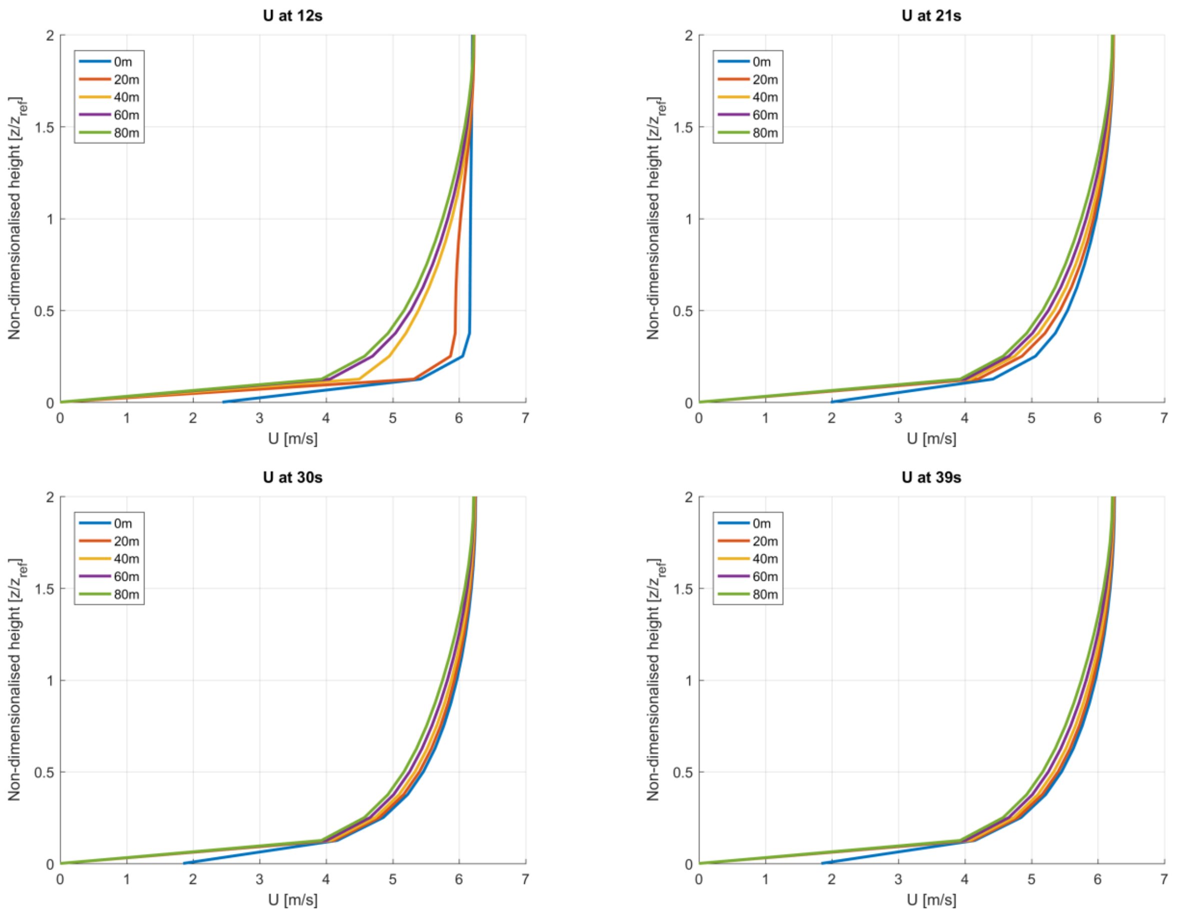


Figure 43. HHTSL velocity profile evaluated at different timesteps, sampled throughout the mesh

The velocity profile stabilizes within 30 seconds of simulation time.

The simulation was stable throughout the mesh after a simulation time of 30 seconds. When the mesh is stabilized, one can take data from the ALM. The one rotation of the turbine takes $0.29s$. One should average approximately ten rotations after the simulation stabilizes, i.e. adding four seconds of simulation time for this.

Based on these results, and comparing to [43], a base mesh of total size $100m\ x\ 100m\ x\ 20m$ with cell length $1.25m\ x\ 1.25m\ x\ 1.25m$, in x-, y- and z direction respectively, was chosen. The total simulation time was set to 34 seconds.

6.2 Time step- and grid sensitivity analysis

This section will show how the time step has been chosen and two methods of spatial grid sensitivity analysis has been conducted to see how the performance curve of the ALM is affected by different parameters. To be noted, when mentioning the power coefficient (C_P), if nothing else is described the C_P refers to the value that is outputted from the model.

At the start when getting to know the model the timestep was kept small to ensure a stable Courant number and convergence of the model. According the reference paper [5], approximately 200 steps per revolution would be sufficient. If one recalls the Equation 3.52 with the constants $TSR = 1.9$, $R = 0.5m$ and $U_\infty = 5.7\ m/s$, the rotor angular velocity is $\omega = 21.66\ rad/s$. The time for one rotation is equal to:

$$t = \frac{2\pi}{\omega} = \frac{2\pi}{21.66\frac{rad}{s}} = 0.29s$$

A very conservative timestep was chosen to $\Delta t = 0.001$, so a total of 290 steps each revolution.

6.2.1 First spatial grid sensitivity analysis, changing *blockMesh*

The reference paper [5] have performed a spatial grid resolution sensitivity analysis, but in that case there is no geometry to disturb the flow. So, to test how the model would be affected by the large grid and implemented geometry, a sensitivity analysis was conducted by simply changing the resolution of *blockMesh*.

The analysis was performed with the finished mesh with the fish farm feed barge geometry and refinement around the turbine. The methodology was to increase the number of cells in x and y from 20 to 200, and z from 4 to 40 which would maintain the background mesh cell aspect ratio. The expected result was that the C_P would converge when reaching a cell length inside the refinement around the ALM smaller than $0.07m$, which is the cell length that the C_P of the reference paper [5] started to converge, but that was in water and with $U_\infty = 1\ m/s$, hence the smaller than $0.07m$.

This was done despite of the TKS and velocity would not be approximately equal when using lower number of cells than decided in section 6.1. The results are shown in Figure 44:

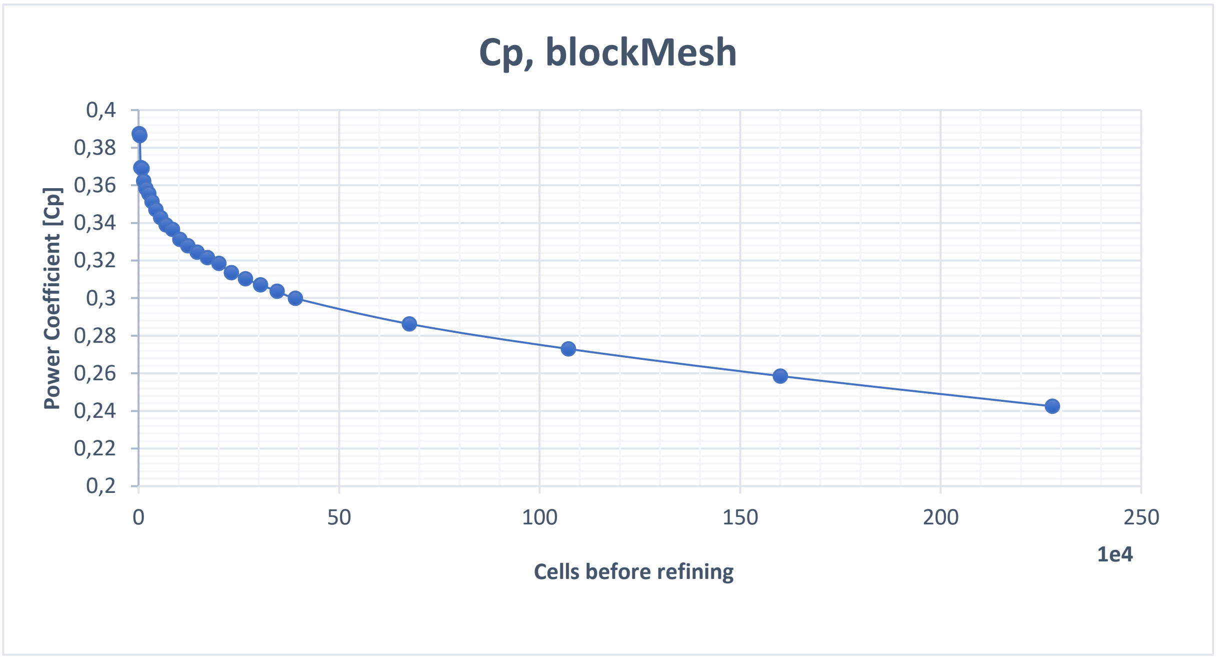


Figure 44. Results from spatial grid sensitivity analysis. The blockMesh has been scaled, while maintaining the aspect ratio of the background mesh. The x-axis represents number of cells in background mesh, in the factor of 1e4.

The results from spatial grid sensitivity analysis was not as suspected. The C_P did not converge, and the trend of the last simulations had almost a linearly negative slope. It looked like that this was directly connected to a field parameter. This was explored using *paraView*, the four last cases were loaded in at the latest timestep. The built-in function *plot over line* was used to plot the different fields of the model at different distances upstream of the turbine, both horizontal and vertical with a line equal to six times the length of the radius of the turbine. No direct connection was found.

The ALM is obviously a more delicate approach and require more than changing of one parameter at same time to converge, i.e. the number of blade elements, time step and Gaussian width parameter. To prepare for a new spatial grid sensitivity analysis, the parameters that was involved were studied. The study is presented in the next section 6.2.2.

6.2.2 Parameter study

The last section showed that the ALM is depending on more than one variable. The more important variables were changed to get performance curve to converge. The parameters and the setup for a new spatial grid sensitivity analysis will be presented in this section.

Number of blade elements

Each blade is divided into a number of elements as discussed in 3.3.1. The number of elements were calculated to approximately equal to the total span divided by the Gaussian width parameter. In this case the Gaussian width that was largest was dependent on mesh size, therefore ϵ_{mesh} is used.

Time step

As explained earlier the timestep should be set according to the size of the cell. In one timestep the model should not rotate more than one cell length. This was adopted, but it did not account for that the blade moves in a circular motion, while in a cartesian grid. The grid and all the turbine placements through one turbine rotation is shown with two examples, one with $\Delta x = 0.0625$ and $\Delta x = 0.03125$, in Figure 45:

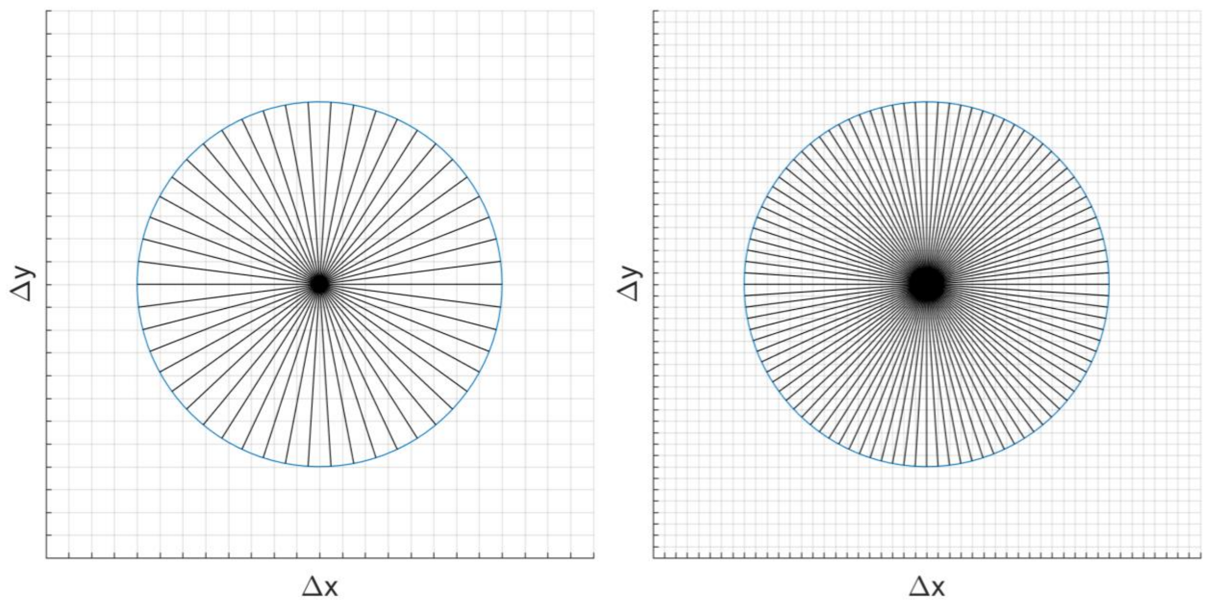


Figure 45. The grid and all the placements through one turbine rotation. $\Delta x = 0.0625$ and $\Delta x = 0.03125$ with $\Delta t = 0058$ and $\Delta t = 0029$, respectively. The blue circle represents the turbine, with $R = 0.5m$ in both cases. The slice is seen from top.

To take a closer look at the periphery a detailed view of the finest grid in Figure 45 is shown in Figure 46.

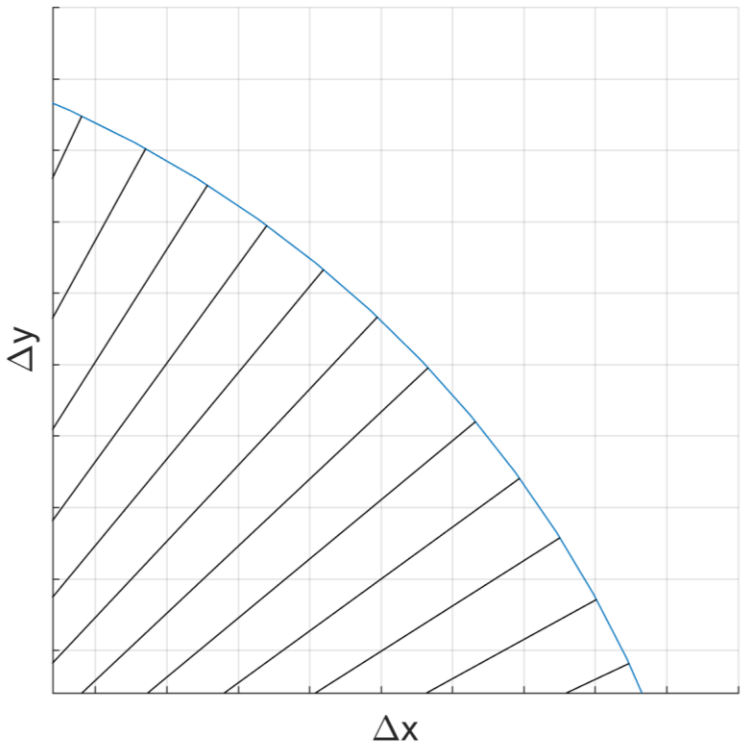


Figure 46. Detailed view of the periphery of the turbine. With $\Delta x = 0.03125$ and $\Delta t = 0.0029$. One can see that the turbine placement occasionally skips over a whole cell.

As seen in Figure 46, when the turbine moves with one cell length each timestep, the turbine occasionally passes through a cell. In these cases, the ALE will not be calculated in this cell, even if it is inside the periphery, but this was to be expected when the turbine moves at the length of one cell each timestep.

With these parameters in mind a new setup for the grid sensitivity analysis was setup, with the formulas from section 3.3.1, as seen in Table 7.

Table 7. Spatial grid sensitivity analysis setup.

$\boldsymbol{blockMesh^3}$	$\boldsymbol{\Delta x}\ [\boldsymbol{m}]$	$\boldsymbol{\epsilon_{mesh}}$	$\boldsymbol{N_{elem,blade}}$	$\boldsymbol{N_{elem,strut}}$	$\boldsymbol{\Delta t}$
2m	0.0625	0.25	4	2	0.0058
1	0.04807	0.192	6	3	0.0044
1.25m	0.0390625	0.15625	7	4	0.0037
1m	0.03125	0.125	8	4	0.0029
0.8m	0.025	0.1	10	5	0.0023

The results from the spatial grid sensitivity analysis with the calculated parameters is shown in Figure 47.

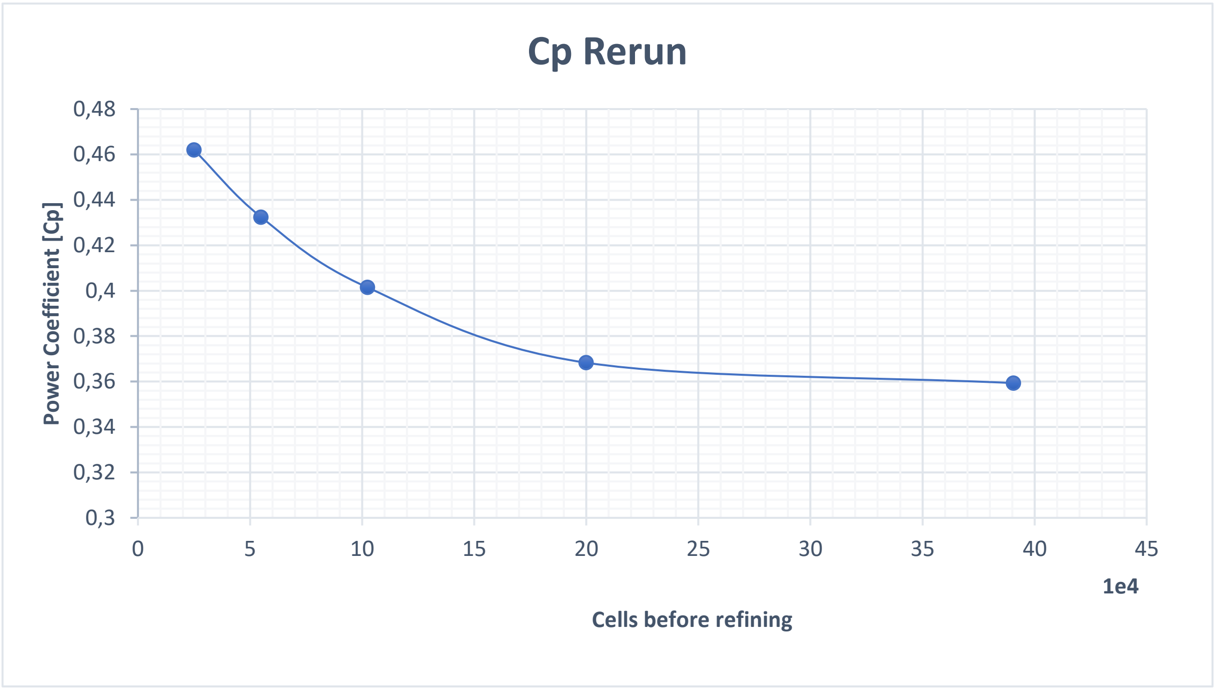


Figure 47. Results from rerun of spatial grid sensitivity analysis. The parameters that have been scaled with the block mesh is Gaussian width, number of elements and the time step. The x-axis represents number of cells in background mesh, in the factor of 1e4.

This spatial grid sensitivity analyses had a much better outcome. The performance curve started to converge when the cell length in the ALM reached $\Delta x = 0.03125$, this was with a background mesh size of $1m\ x\ 1m\ x\ 1m$ and with 100 time steps per revolution. This also shows that with better adapted parameters, the overall performance of the turbine is better, considering this analysis was done with the same inlet conditions as the previous spatial grid sensitivity analysis.

6.3 Model analysis

The model has been inspected, and the different parameters such as the velocity, turbulent kinetic energy and turbine forces has been observed, using the default case which is the tower height of 6m and inlet from North. The simulation was run with the parameters from the results in section 6.1 with the fish farm feed barge geometry implemented. The data was taken from the last time step.

First the tutorial cases in *turbinesFoam* was run to show how both the axial- and cross flow turbines impact on the velocity, the cases has not been modified in any way.

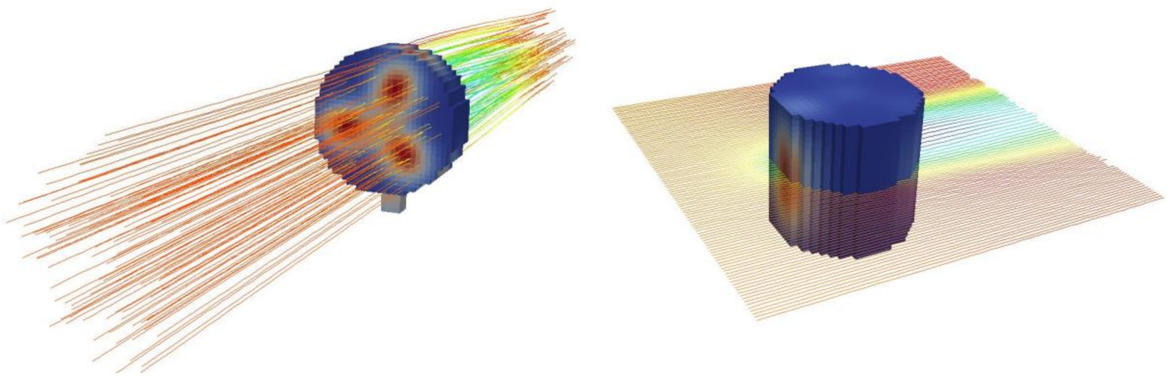


Figure 48. Figures from tutorial cases in turbinesFoam, the axial- and cross flow turbines, respectively. The cases have not been modified. The cellZones is colored by the turbine forces, and the streamlines the velocity.

6.3.1 Velocity profile, K profile and Turbine forces

The figures show the surroundings of the turbine, have been clipped with a box, then sliced. The turbine was extracted with *extractBlock*, then sliced. To better understand why the velocity- and TKE fields have the form which they have, the placement of the turbine blades, which is showed with the turbine forces, is inspected in Figure 49.

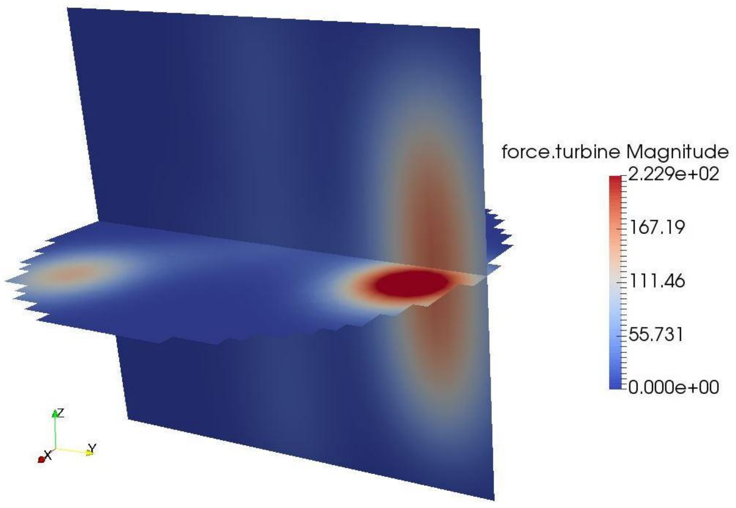


Figure 49. Turbine forces. Sliced turbine in YZ and XY plane.

The velocity and the TKE in the turbine is shown in Figure 50:

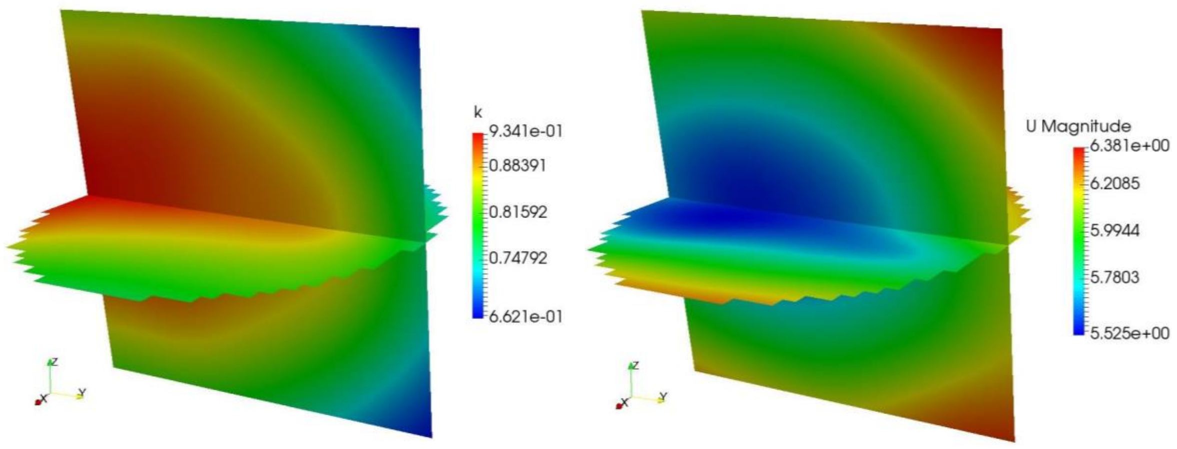


Figure 50. TKE and velocity in the turbine, respectively. Sliced turbine in YZ and XY plane.

From Figure 49 one can see how the spherical Gaussian function smears the forces back onto the grid. The forces are largest when the blade is coming back upwind, due to the higher relative wind velocity and angle of attack. In Figure 50 when can see that the velocity field inside the turbine is lower, which is typical for VAWT's, especially right after the blade. As for the turbulent kinetic energy, one can clearly see that it increases inside the turbine. To see how the fields surrounding the turbine were affected, slices are presented with the turbine *cellZone* and samples horizontally upstream and downstream was performed.

Turbulent kinetic energy cuts are shown in Figure 51:

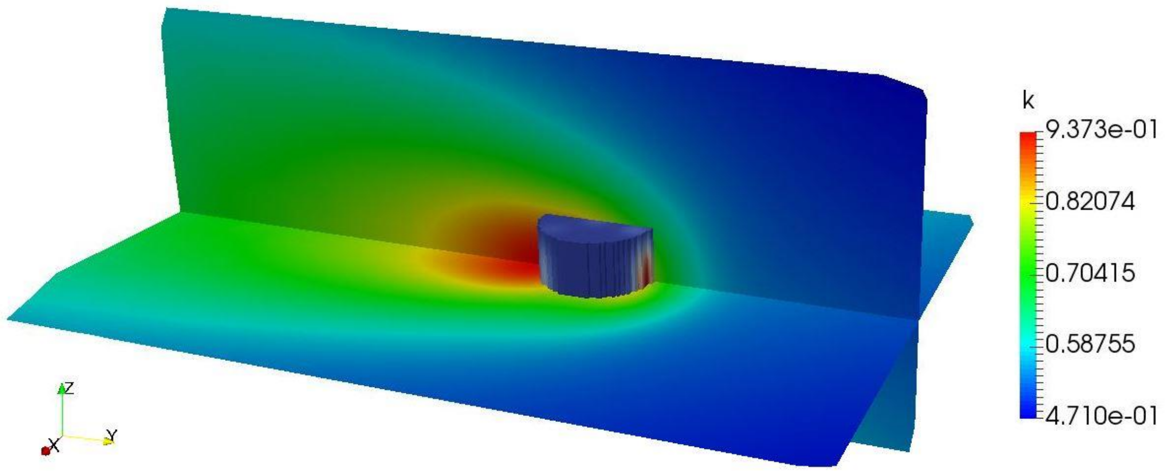
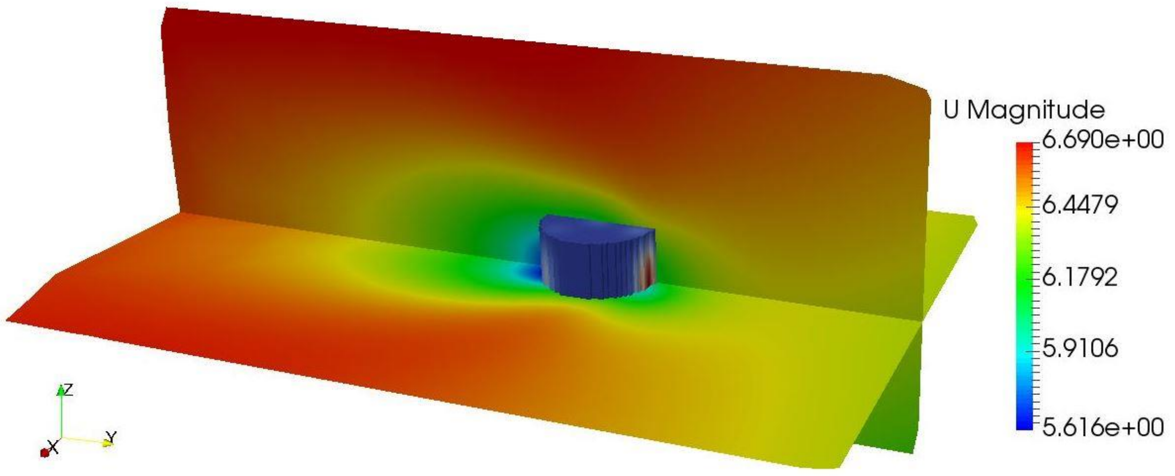


Figure 51. Turbulent kinetic energy in the YZ and XY plane.

The velocity cuts are shown in Figure 52:



Turbulent kinetic energy and velocity sampled horizontally upstream and downstream is shown in Figure 53:

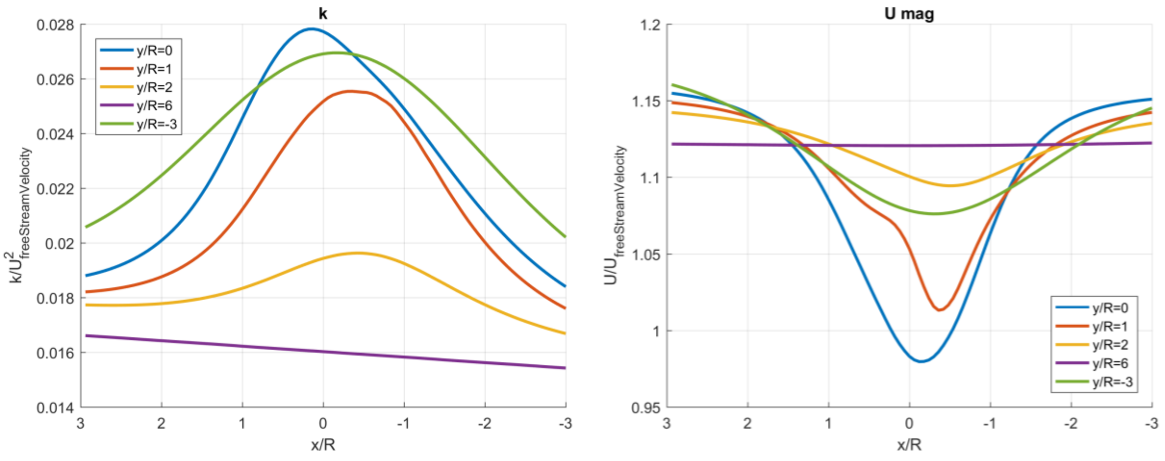


Figure 53. Turbulent kinetic energy and velocity sampled in the x-horizontal direction, respectively.

These results show how the different parameters are affected by the turbine. The velocity recovers shortly after the turbine, this is also due to the velocity increase because of the fish farm feed barge geometry. From the velocity samples in Figure 53 one can see that the *freeStreamVelocity* is lower than the wind velocity upstream of the turbine, and the drop-in velocity at $y/R = 1$ is due to turbine blade coming back upstream. From the turbulent kinetic energy sample in Figure 53, the distinctive “drop” of TKE in the center of the turbine is not absence, which can be seen in the refrence paper [5], this could be due to the grid resolution of the ALM.

6.4 Tallaksholmen

In this section, the framework that has been built during this thesis project will be presented by showing the results of optimization of the turbine height. A comparison between empirical and simulation, regarding the wind speed and power output has been conducted. The framework that has been developed has been tested on completely different fish farm feed barge, to show the flexibility of the model.

There was conducted multiple simulations of Tallaksholmen during this work, mainly two sets; one set of simulations which was presented in April at SOWE 2017, then another set when the model parameters was set after the second spatial grid sensitivity analysis. Both sets results will be presented.

The first set of simulation was run from two directions, North and South, and at three different heights. The second set of simulations were run from four directions, North, South, East and West, and at three different heights.

6.4.1 Torque reduction, Power output, Height- and Direction change

The different heights that were inspected was set to 4m, 6m and 10m. These different heights

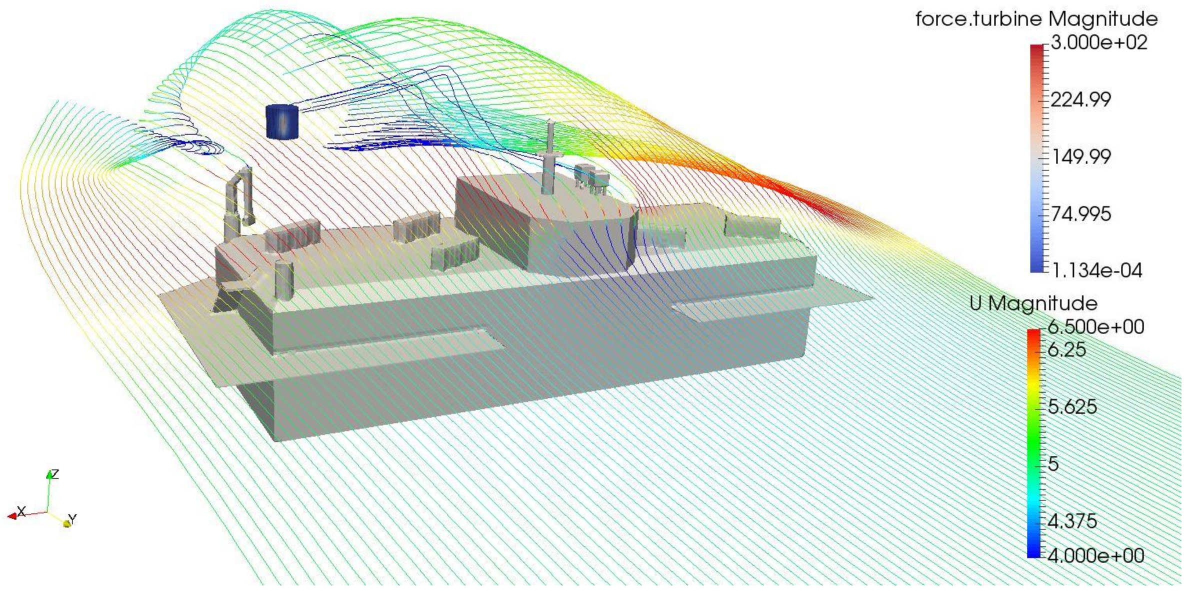


Figure 54. Tallaksholmen with streamlines, seeded at 5m height. The figure shows the typical flow field over the geometry. Turbine height is 6m.

were decided, as these are the tower heights available from the wind turbine manufacturer. Since the tower is set on the fish farm feed barge with the fundament located at 8m height, the center of the wind turbine is set to 12m, 14m, and 18m. For reasons discussed in 3.3.4, the C_P that is outputted form the model was not used, but the calculated torque has been. From the torque, the power output of the turbine has been calculated.

Before the results are presented and discussed, figures from the simulations are presented.

The three different heights are shown in Figure 55:

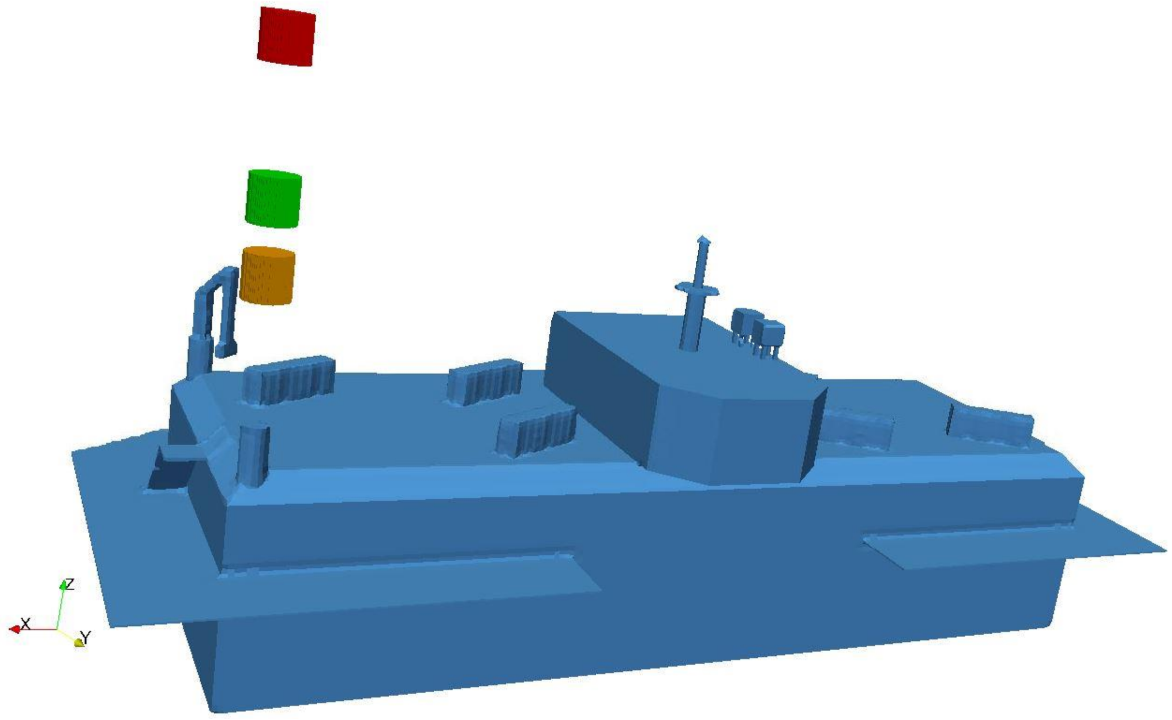


Figure 55. Figure from simulation showing the three different tower heights. 4m, 6m and 8m which represents 12m, 14m, and 18m, respectively in the simulation coordinate system.

Streamlines with velocity coloring for each height is shown in Figure 56:

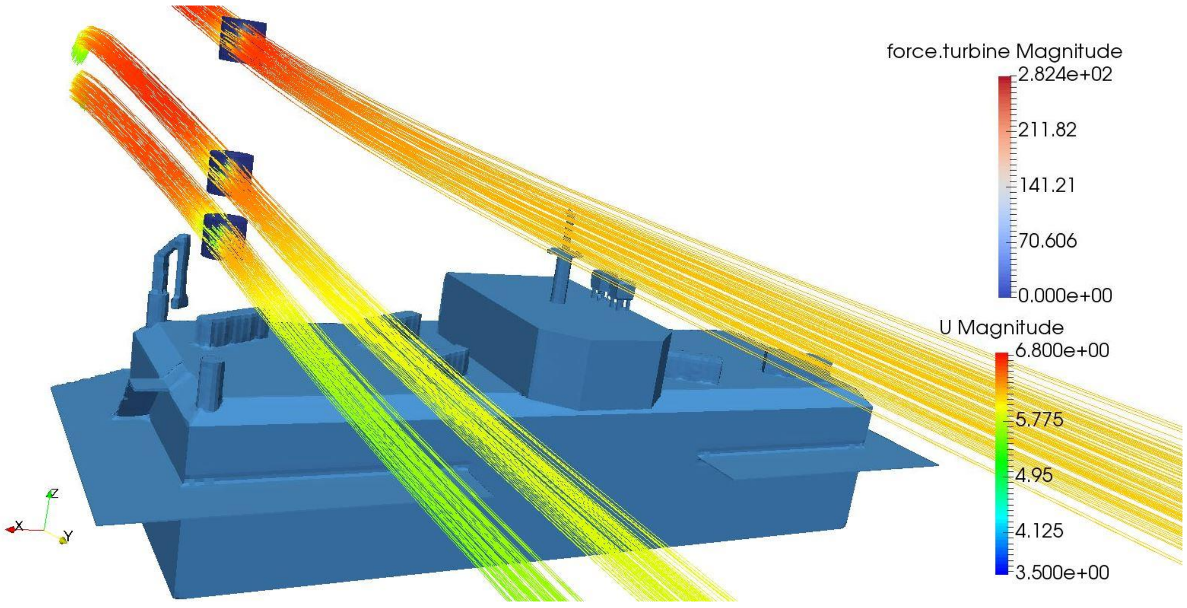


Figure 56. Streamlines with the use of point source with radius 0.6m at each height. The internal mesh has been extracted from each case. The cases showing is from the second set of simulations with wind from North.

Velocity field with cut in YZ direction, seen from East is shown in Figure 57:

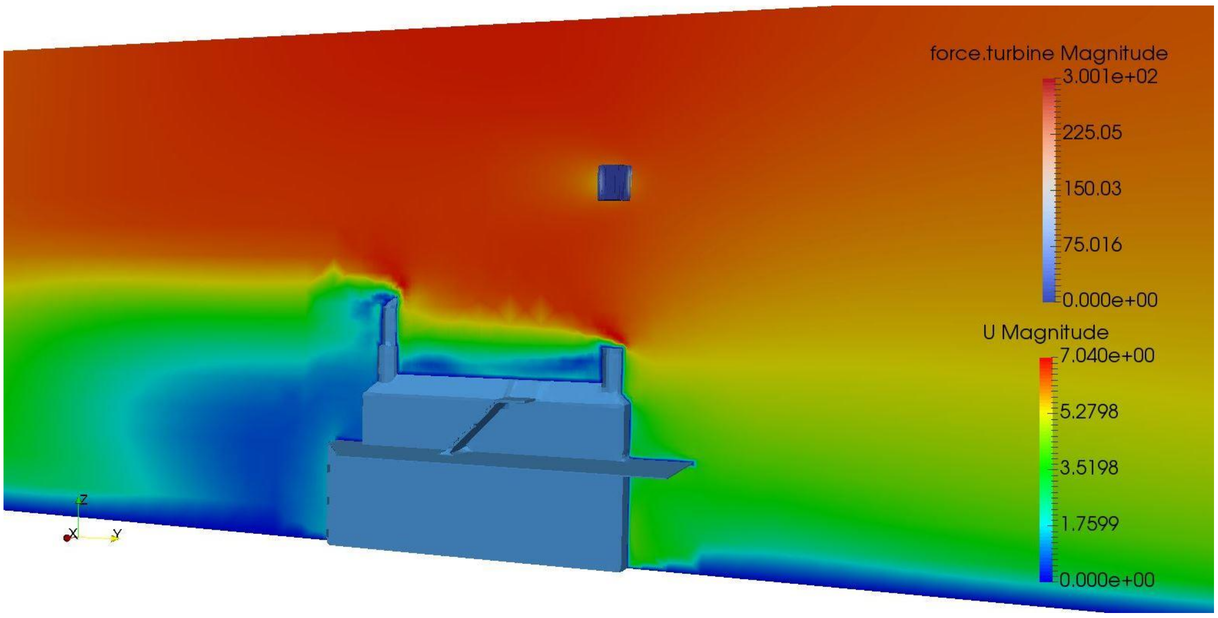


Figure 57. Velocity field with cut in YZ direction, seen from East. The turbine tower height is 6m, inlet from North.

Streamlines with inlet from West, this case gave the worst power output is shown in Figure 58:

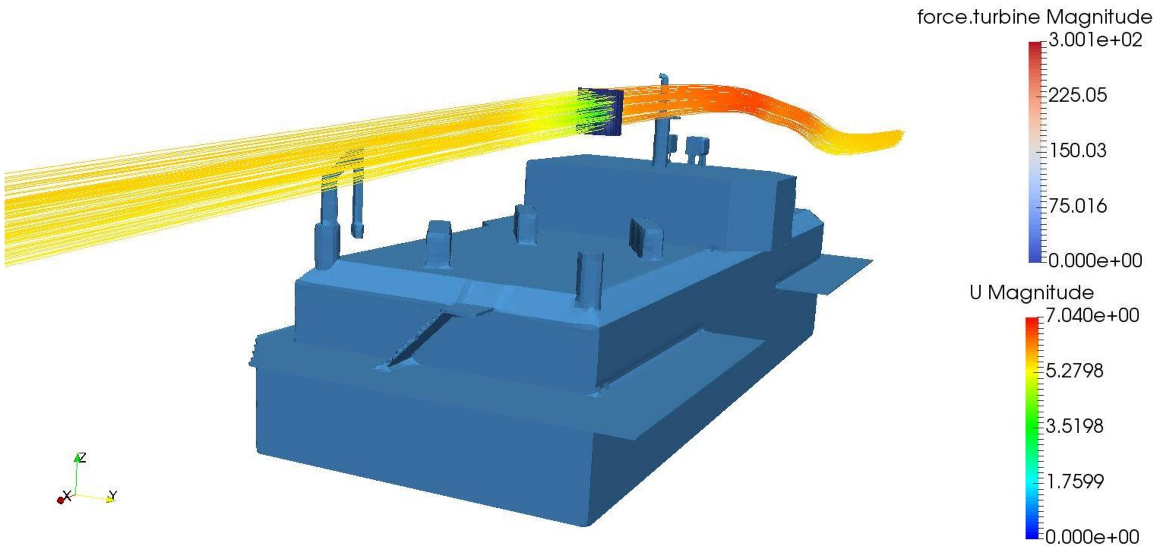


Figure 58. Tower height is 4m, with inlet from West. This simulation gave the lowest torque, as expected. This was expected considering flow field is more disturbed than in the other cases.

In the first set of simulations, the mean torque and power with inlet from North and South has been used. In the second set of simulations, the mean torque and power with inlet from all the sided; North, South, East and West has been used. On the seconds set of simulations two results are presented, one set of result presents only the mean from all sides, then one result with corrections from the wind rose. When the correction from the wind rose was done, the percentage of each wind direction has been considered.

The result from the torque and power output is given in Table 8:

Table 8. Mean results from first-, second set of simulations and with the second set with correction from the wind rose.

Torque [Nm]

	First set	Second set	Second set, corrected with wind rose
4m	1,527	1,561	1,540
6m	1,744	1,763	1,766
10m	1,913	1,990	1,982

Power Output [W]

4m	33,08	33,80	33,35
6m	37,78	38,19	38,25
10m	41,43	45,02	44,53

The result from power output in Table 8 shown graphicly in Figure 59:

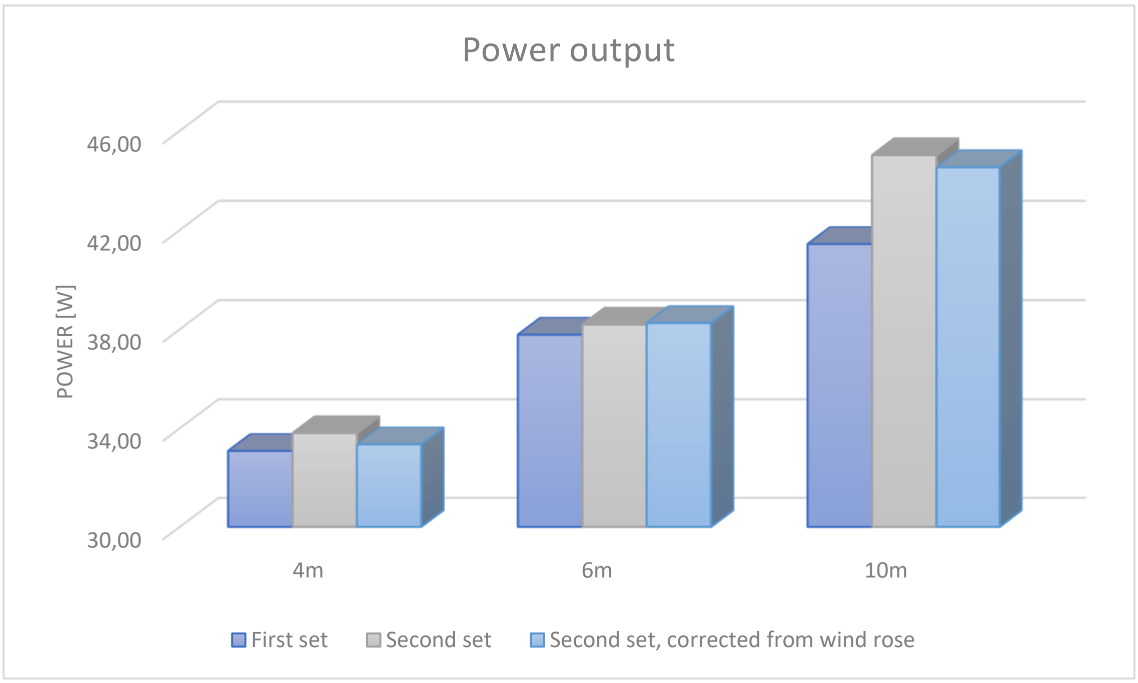


Figure 59. Power output results shown graphicly. This also represent the torque only scaled with the angular velocity

The torque reduction from the different heights is given in Table 9:

Table 9. Results torque reduction, presented in percentage.

Torque Reduction [%]

	4m	6m	10m
First set	79,8 %	91,2 %	100,0 %
Second set	78,4 %	88,6 %	100,0 %
Second set, corrected with wind rose	77,7 %	89,1 %	100,0 %

The results show that with the first set of simulations, the overall performance was worse than the second set. The torque reduction however is not as high, this was to be expected considering the flow inlet were only from North and South, where the flow was not that disturbed.

The second set showed overall better performance. This was due to the tuning of parameters. Since the second set was run with inlets from all directions, the results should show a better approximation of the reality. Considering the wind rose as well should even better the approximation.

The results that has been presented will be used for a cost analysis in section 6.5.

6.4.2 Empirical versus Simulation

There has been conducted a comparison between the wind velocity in the results from the simulation i.e. the default case with 6m tower height and wind from North, and the logarithmic law formulations discussed in section 3.2.8. The velocities from the simulation was sampled along the YZ plane in the cut as shown in Figure 57. The samples were taken from the top-down direction. The velocity profiles sampled together with the logarithmic law formulation is shown in Figure 60:

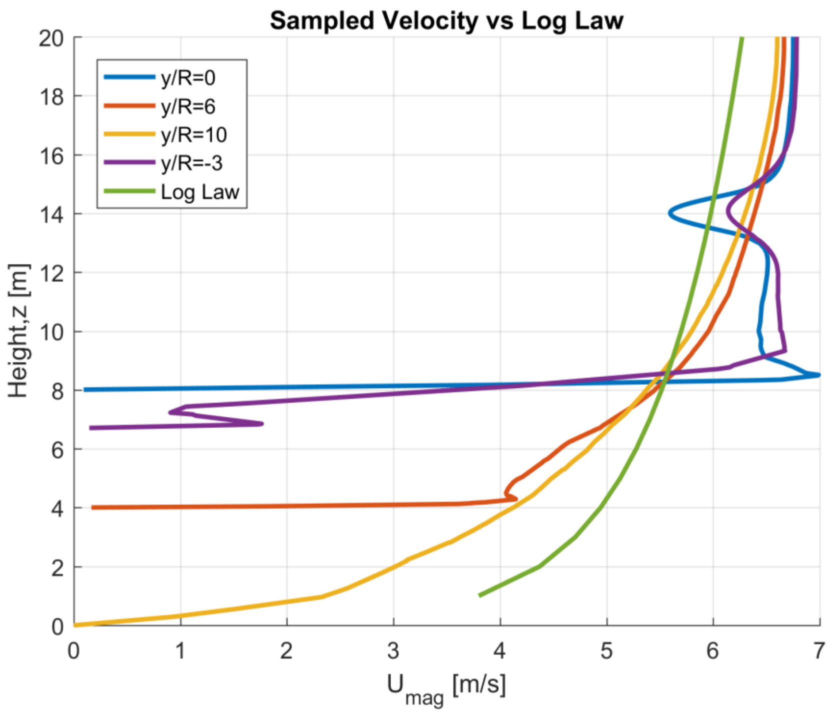


Figure 60. Sampled velocity compared with logarithmic law formulation.

The result showed, as one can see in Figure 60, that the velocity is accelerating due to the geometry. This lead to a higher velocity upwind of the turbine in the simulations than the logarithmic law function, which in this case would lead to a better performance.

6.4.3 Nordheim

To show the flexibility of the framework for the model, a second fish farm feed barge has been inspected. In this case only wind from one direction has been considered. The geometric model was based purely on pictures from Gwind, the model was not exact but it did represent the geometry well.

Table 10. Results from Nordheim.

	Nordheim
Torque [Nm]	*2,159*
Power output [W]	*46,78*

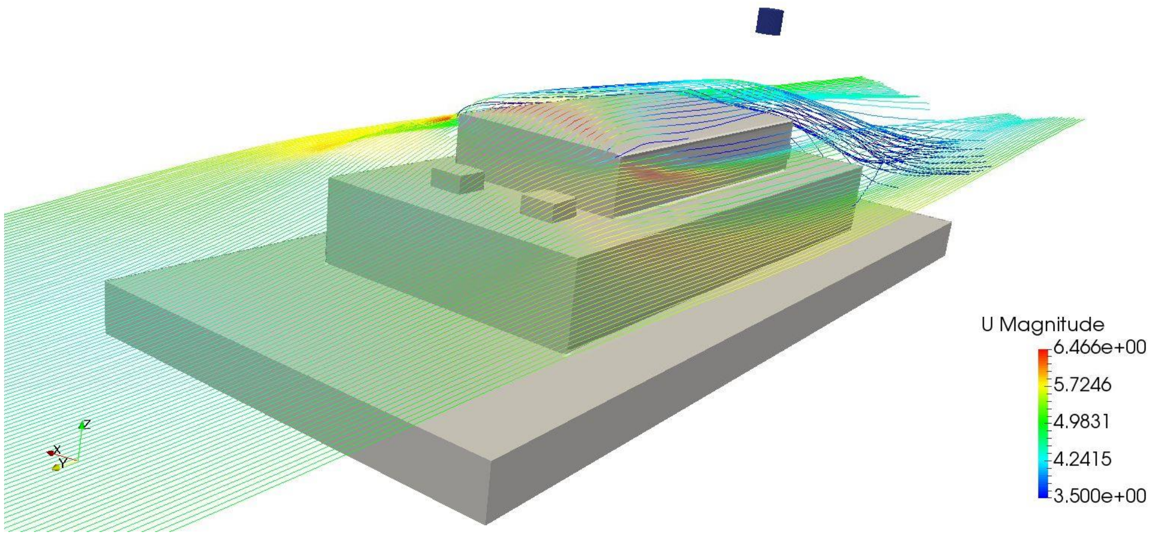


Figure 61. Nordheim fish farm feed barge.

First the local wind was inspected by turning off the ALM, so the model could be run faster just to get an indication of where the velocity is high. The turbine was placed in high velocity zone. The power output result seen in Table 10, was good.

This case used the same inlet conditions as the second set. The time spent setting up a new case, with the help of the framework, took approximately three hours, that includes the geometric modeling as well.

6.5 Cost Consideration Tallaksholmen

A cost analysis has been conducted, this shows how the model could potentially be used for practical engineering problems. In this case the power output results from the second set with the wind rose correction on Tallaksholmen fish farm feed barge has been investigated. The cost consideration where evaluated dependent on; cash flow, net present value (NPV) and the investment cost. Also, one need to consider that on Tallaksholmen the tower has to be lowered each time the food silos are filled.

This analysis was based on wind turbine with the capacity of $3.2kW$. The parameters used for the cost analysis was: a plant load factor of 30%, discount factor of 7%, maintenance cost of 3% of the investment, price per kWh of $4\ NOK$. The cash flow was investigated over 20 years.

The investment cost was set in collaboration with Gwind.

6.5.1 Cash flow, NPV and Investment cost

The result from the cost analysis is shown in Figure 62:

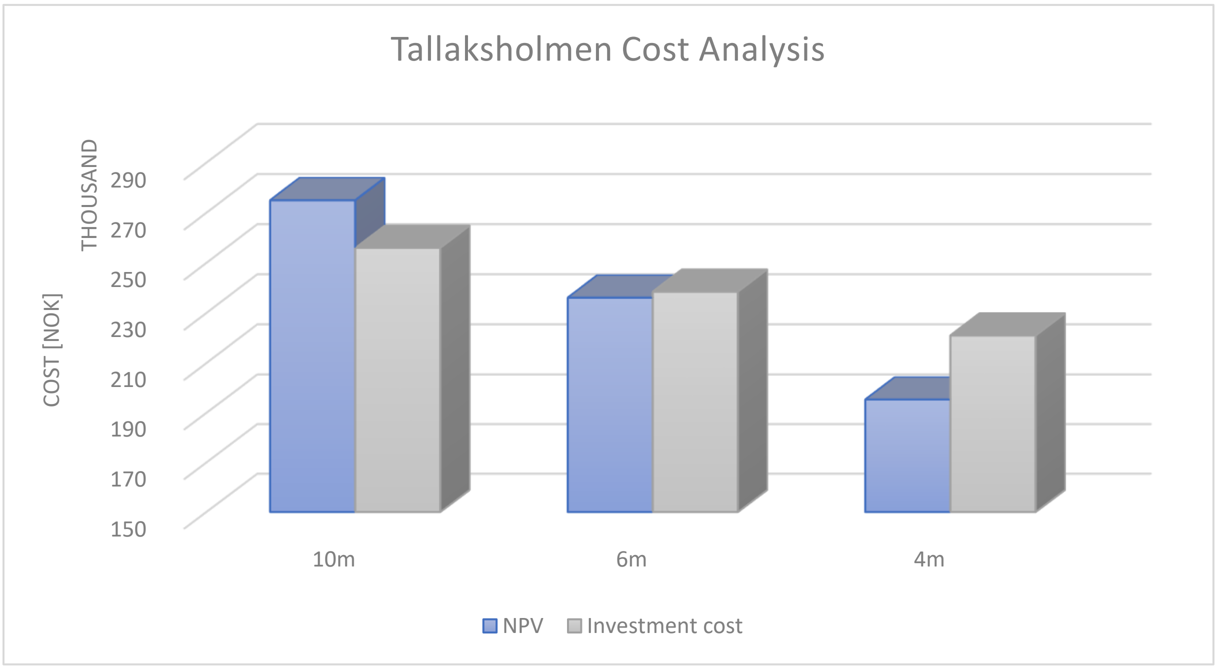


Figure 62. Tallaksholmen cost analysis. Investment cost and NPV has been investigated with the results from the second set of simulations, also considering the correction from the wind rose.

The results show that the best investment considering cost would be the be the 10m tower height. Considering that one must handle the turbine each time the food siloes are refilled, a lower tower would be preferable. In that case the 6m tower would be the best option. Where the power production is only 11% less than the 10m option, whereas the 4m option produces 22% less.

7 Conclusion and Future work

The main objectives of this thesis were to develop a framework for wind flow simulations over fish farm feed barges, which have been done.

Measurements from Tallaksholmen were collected and analyzed in order to establish local wind conditions for the actual site. By utilizing technical drawings of the fish barge made available from Grieg Sea Food, a 3D model of the barge was created. A STL surface of the feed barge was exported from the 3D model and imported to OpenFOAM toolbox.

A framework has been made, and were used on two different cases of fish farm feed barges. The framework includes an actuator line model of a vertical axis wind turbine, simulated with OpenFOAM's *pimpleFoam* solver, with the *k-epsilon* turbulence model. The inlet has been enriched with an atmospheric boundary layer, a grid dependency analysis was conducted and resulted in base mesh of size $100m\ x\ 100m\ x\ 20m$ with cell length $1.25m\ x\ 1.25m\ x\ 1.25m$ and run time of $34s$ ensuring a steady solution.

Two spatial grid sensitivity analysis has been performed on the ALM. The first, where only the *blockMesh* was changed, the performance curve did not converge. The second, after a parameter study the performance curve converged, which indicates that the ALM is a delicate model depending on multiple parameters at once.

Lastly, a brief analysis was conducted in order establish a cost optimal relationship between turbine height, material cost and power production. This will further be useful for Gwind and Grieg Sea Food in their project development.

The framework was also used to investigate the optimal height correlated with cost benefit, which resulted in tower height of $6m$. This showed that the model could potentially be used in a more practical engineering approach.

The flexibility of the framework was demonstrated by introducing a completely different fish farm feed barge, setup and ready to run in a few hours. The case was run first without the ALM in order to investigate the local wind flow, then rerun the case with the turbine placed in the high velocity zone, which gave good results.

7.1 Future work

This work will hopefully act as a stepping stone to further investigating wind flow simulations over fish farm feed barges, considering floating turbines and implementation of multiple turbines to construct a small turbine park.

As discussed in section 3.3.4, initially the plan was to change the source code so that the *freeStreamVelocity,* would be sampled at each iteration so the performance coefficients would correctly outputted. The author of *turbinesFoam* was contacted regarding this issue, and it was decided that would require more elaborate studies and analysis.

Furthermore, the ALM has the ability to scale the turbine, this could be investigated. It also contains low fidelity numerical modeling for the horizontal axis wind turbine, which was only tested, but should be compared with a more validated ALM e.g. NREL actuator line modeling library SOWFA.

If a VAWT is installed on Tallaksholmen which is recommended, the operational performance should be compared against the simulations to further verify the computational approach.

8 References

[1] Kartverket.no. (2017, 23.02). *Fishfarms*. Available: https://kart.fiskeridir.no/

[2] Gwind. (2017, 20.02). *Gwind*. Available: http://www.gwind.no/

[3] H. L. Syse, "Investigating Off-Grid Energy Solutions for the Salmon Farming Industry," Master, University of Strathclyde and Universitetet i Stavanger, 2016.

[4] P. Bachant, A. Goude, and M. Wosnik. (2016, 23.02). *turbinesFoam: v0.0.7*. Available: https://doi.org/10.5281/zenodo.49422

[5] P. Bachant, A. Goude, and M. Wosnik, "Actuator line modeling of vertical-axis turbines," *arXiv preprint arXiv:1605.01449,* 2016.

[6] L. Riva, "Implementation and Application of the Actuator Line Model by OpenFOAM for a Vertical Axis Wind Turbine set on a University of Stavanger building roof," Master thesis, University of Stavanger and Politecnico di Milan, Unpublished, 2017.

[7] UiB. (2017, 16.03). *Band 5: Havbruk*. Available: https://norges-fiskeri-og-kysthistorie.b.uib.no/bokverket/bind-5-havbrukshistorie/

[8] SSB. (2017, 16.03). *Toppår for oppdrettslaks*. Available: http://www.ssb.no/jord-skog-jakt-og-fiskeri/statistikker/fiskeoppdrett/aar/2016-10-28

[9] N.-o. fiskeridepartementet. (2017, 16.03). *Aquaculture*. Available: http://www.fisheries.no/aquaculture#.WMp-iPk1-Uk

[10] Seafood Grieg, "Tinnlandet," T. j. q1 En flott dag pÃ¥, Ed., ed. Grieg Seafood, 2017.

[11] G. W. o. Norway, "Report on the Environmental Impact of farming of North Atlantic Salmon in Norway," *Environmental facts of Norwegian salmon farming,* 08.2010 2011.

[12] M. Dumiak, "Lice-Hunting Underwater Drone Protects Salmon With Lasers," 2017.

[13] S. M. Solutions. (2017, 22.05). *Stingray*. Available: http://www.stingray.no/

[14] M. Porter, N. Duncan, D. Mitchell, and N. Bromagea, "The use of cage lighting to reduce plasma melatonin in Atlantic salmon (Salmo salar) and its effects on the inhibition of grilsing," *Aquaculture,* vol. 176, no. 3, pp. 237-244, 1999.

[15] M. Harvest, Salmon Farmin Industry Handbook 2016, Marineharvest.com, 2016. [Online]. Available: http://marineharvest.com/globalassets/investors/handbook/2016-salmon-industry-handbook-final.pdf.

[16] A. Group, "Cage Farming," ed. akvagroup.com, 2016.

[17] S. NASSAUER. (2014, 22.03). *Top Chefs, Grocers Choose Farmed Salmon*. Available: https://www.wsj.com/articles/top-chefs-grocers-choose-farmed-salmon-1411600146

[18] J. F. Manwell, J. G. McGowan, and A. L. Rogers, *Wind energy explained: theory, design and application*. John Wiley & Sons, 2010.

[19] NREL. (2017, 13.05). *11.3 Technology characterization | Global CCS Institute*. Available: https://hub.globalccsinstitute.com/publications/renewable-electricity-futures-study-volume-2-renewable-electricity-generation-and-storage-technologies/113-technology-characterization#fig_11-5

[20] J. M. Baskin, G. E. Miller, and W. Wiesner, "Wind turbine rotor," ed: Google Patents, 1985.

[21] E. Muljadi and C. P. Butterfield, "Pitch-controlled variable-speed wind turbine generation," *IEEE transactions on Industry Applications,* vol. 37, no. 1, pp. 240-246, 2001.

[22] J. R. Cotrell, "A preliminary evaluation of a multiple-generator drivetrain configuration for wind turbines," in *ASME 2002 Wind Energy Symposium*, 2002, pp. 345-352: American Society of Mechanical Engineers.

[23] S. Sheng and P. S. Veers, *Wind turbine drivetrain condition monitoring-an overview*. National Renewable Energy Laboratory Golden, Colo, USA, 2011.

[24] W. Musial, S. Butterfield, and B. McNiff, "Improving wind turbine gearbox reliability," in *European Wind Energy Conference, Milan, Italy*, 2007, pp. 7-10.

[25] S. Muller, M. Deicke, and R. W. De Doncker, "Doubly fed induction generator systems for wind turbines," *IEEE Industry applications magazine,* vol. 8, no. 3, pp. 26-33, 2002.

[26] J. Jonkman, S. Butterfield, W. Musial, and G. Scott, "Definition of a 5-MW reference wind turbine for offshore system development," *National Renewable Energy Laboratory, Golden, CO, Technical Report No. NREL/TP-500-38060,* 2009.

[27] M. Karimirad and T. Moan, "Wave-and wind-induced dynamic response of a spar-type offshore wind turbine," *Journal of waterway, port, coastal, and ocean engineering,* vol. 138, no. 1, pp. 9-20, 2011.

[28] M. O. Hansen, *Aerodynamics of wind turbines*. Routledge, 2015.

[29] A. Muratoglu and M. I. Yuce. (2014, 03.05). *ARE - Performance Analysis of Hydrokinetic Turbine Blade Sections*. Available: http://are.avestia.com/2015/001.html

[30] D. o. P. S. U. Robert B. Laughlin. (2007, 03.05). *Lift*. Available: http://large.stanford.edu/courses/2007/ph210/glownia2/

[31] P. Jain, *Wind energy engineerin*. McGraw-Hill, 2011.

[32] T. Burton, D. Sharpe, N. Jenkins, and E. Bossanyi, *Wind energy handbook*. John Wiley & Sons, 2001.

[33] M. Ragheb and A. M. Ragheb, *Wind turbines theory-the betz equation and optimal rotor tip speed ratio*. INTECH Open Access Publisher, 2011.

[34] F. Moukalled, L. Mangani, and M. Darwish, *The finite volume method in computational fluid dynamics*. Springer, 2016.

[35] H. K. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education, 2007.

[36] Y. A. Cengel and J. M. Cimbala, *Fluid mechanics, Fundamental and applications*. McGraw Hill, New York, 2006.

[37] OpenCFD, User Guide, 2017. [Online]. Available: https://sourceforge.net/projects/openfoamplus/files/v1612+/UserGuide.pdf. Accessed on 13.05.2017.

[38] Cener. (2017, 13.05). *SOWE 2017 | Windbench*. Available: http://windbench.net/sowe-2017

[39] O. Reynolds, "On the dynamical theory of incompressible viscous fluids and the determination of the criterion," *Proceedings of the Royal Society of London,* vol. 56, no. 336-339, pp. 40-45, 1894.

[40] D. B. Spalding, "A Single Formula for the "Law of the Wall"," Accessed on: 25.05.2017. doi: 10.1115/1.3641728 Available: http://appliedmechanics.asmedigitalcollection.asme.org/article.aspx?articleid=1394697

[41] OpenCFD. (2017, 04.06). *OpenFOAM: Residuals*. Available: http://openfoam.com/documentation/cpp-guide/html/guide-solvers-residuals.html#sec-solvers-residuals-tolerances

[42] E. P. Poeter. (2017, 04.06). *The Courant Number*. Available: http://inside.mines.edu/~epoeter/583/13/discussion/courant.htm

[43] D. M. Hargreaves and N. G. Wright, "On the use of the k-epsilon model in commercial CFD software to model the neutral atmospheric boundary layer," *Wind Engineering and Industrial Aerodynamics,* vol. 95, pp. pp 335-369, 2007.

[44] OpenFoam. (2017, 04.06). *OpenFOAM: atmBoundaryLayer Class Reference*. Available: https://cpp.openfoam.org/v3/a00073.html#details

[45] D.-D. N. Veritas, "DNV-OS-J101 offshore standard," *Design of offshore wind turbine structures,* 2010.

[46] G. DNV, "DNV-RP-C205: Environmental conditions and environmental loads," *DNV GL, Oslo, Norway,* 2014.

[47] S. E. Nuland, "turbinesFoam, Email correspondence," P. Bachant, Ed., ed, 2017.

[48] H. Ø. Lewis, "- Se, den flyter," 2013.

[49] H. Audun, "Kranglet om fargen i en måned, men nå kommer vindturbinen," 2017-05-18 2017.

[50] G. Seafood. (2017, 23.02). *Grieg Seafood Rogaland*. Available: http://www.griegseafood.no/produksjon/grieg-seafood-rogaland-gsfr/

[51] S. A. Harald. (2011, 05.05.2017). *Hordaland på børs*. Available: https://bergen-chamber.no/visageimages/Pdf_files/180811_Grieg_Seafood.pdf

[52] OpenCFD. (2017, 04.06). *OpenFOAM: Schemes*. Available: http://openfoam.com/documentation/cpp-guide/html/guide-schemes.html

[53] OpenCFD. (2017, 04.06). *OpenFOAM: applications/solvers/incompressible/pimpleFoam/pimpleFoam.C File Reference*. Available: http://openfoam.com/documentation/cpp-guide/html/pimpleFoam_8C.html#details

9 Appendix

9.1 List of figures

9.2 List of tables

9.3 Log files

9.3.1 checkMesh

```
/*---------------------------------------------------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
Build  : 3.0.x-4709bde9acf0
Exec   : checkMesh
Date   : Jun 03 2017
Time   : 17:13:14
Host   : "gorina1.ux.uis.no"
PID    : 161168
Case   : /nfs/student/sveinen/height_change/West/W_6m
nProcs : 1
sigFpe : Enabling floating point exception trapping (FOAM_SIGFPE).
fileModificationChecking : Monitoring run-time modified files using timeStampMaster
allowSystemOperations : Allowing user-supplied system call operations

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
Create time

Overriding DebugSwitches according to controlDict
    crossFlowTurbineALSource 0;

    actuatorLineSource 0;

    actuatorLineElement 0;

    LeishmanBeddoes 0;

    LeishmanBeddoes3G 0;

Create polyMesh for time = 0

Time = 0

Mesh stats
    points:           824357
    faces:            2291027
    internal faces:   2218267
    cells:            733735
    faces per cell:   6.14567112105
    boundary patches: 7
    point zones:      0
    face zones:       0
    cell zones:       1

Overall number of cells of each type:
    hexahedra:     692709
    prisms:        3035
    wedges:        0
    pyramids:      0
    tet wedges:    1
    tetrahedra:    0
    polyhedra:     37990
    Breakdown of polyhedra by number of faces:
        faces   number of cells
            4   30
            5   419
            6   4579
            9   29570
           12   2968
           15   402
           18   22

Checking topology...
    Boundary definition OK.
    Cell to face addressing OK.
    Point usage OK.
    Upper triangular ordering OK.
    Face vertices OK.
    Number of regions: 1 (OK).

Checking patch topology for multiply connected surfaces...
                   Patch    Faces   Points                  Surface topology
                   inlet     1280     1377  ok (non-closed singly connected)
                  outlet     1280     1377  ok (non-closed singly connected)
                  ground     7780     8456  ok (non-closed singly connected)
                   walle     1280     1377  ok (non-closed singly connected)
                   wallw     1280     1377  ok (non-closed singly connected)
                     top     6400     6561  ok (non-closed singly connected)
             Floater_STL    53460    53896  ok (non-closed singly connected)

Checking geometry...
    Overall domain bounding box (-50 -50 0) (50 50 20)
    Mesh has 3 geometric (non-empty/wedge) directions (1 1 1)
    Mesh has 3 solution (non-empty) directions (1 1 1)
    Boundary openness (-4.90553594494e-17 -1.79076107669e-16 -8.55008978989e-17) OK.
    Max cell openness = 3.50187801321e-16 OK.
    Max aspect ratio = 2.87527306357 OK.
    Minimum face area = 0.00152585006567. Maximum face area = 1.60490784807.  Face area
magnitudes OK.
    Min volume = 5.96038026682e-05. Max volume = 2.01682142456.  Total volume = 197544.952417.
Cell volumes OK.
    Mesh non-orthogonality Max: 62.404788765 average: 6.67807950787
    Non-orthogonality check OK.
    Face pyramids OK.
    Max skewness = 1.00418263754 OK.
    Coupled point location match (average 0) OK.

Mesh OK.

End
```

9.3.2 topoSet

```
/*---------------------------------------------------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
Build  : 3.0.x-4709bde9acf0
Exec   : topoSet -parallel
Date   : Jun 03 2017
Time   : 16:15:58
Host   : "gorina1.ux.uis.no"
PID    : 143002
Case   : /nfs/student/sveinen/height_change/West/W_6m
nProcs : 1
Pstream initialized with:
    floatTransfer      : 0
    nProcsSimpleSum    : 0
    commsType          : nonBlocking
    polling iterations : 0
sigFpe : Enabling floating point exception trapping (FOAM_SIGFPE).
fileModificationChecking : Monitoring run-time modified files using timeStampMaster
allowSystemOperations : Allowing user-supplied system call operations

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
Create time

Overriding DebugSwitches according to controlDict
    crossFlowTurbineALSource 0;

    actuatorLineSource 0;

    actuatorLineElement 0;

    LeishmanBeddoes 0;

    LeishmanBeddoes3G 0;

Create polyMesh for time = 0
```

```
Reading topoSetDict

Time = 0
    mesh not changed.
Created cellSet turbine
    Applying source cylinderToCell
    Adding cells with centre within cylinder, with p1 = (14.3 5.3 13.375), p2 = (14.3 5.3 14.625)
and radius = 0.625
    cellSet turbine now size 25696
Created cellZoneSet turbine
    Applying source setToCellZone
    Adding all cells from cellSet turbine ...
    cellZoneSet turbine now size 25696

End

Finalising parallel run
```

9.3.3 blockMesh

```
/*---------------------------------------------------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
Build  : 3.0.x-4709bde9acf0
Exec   : blockMesh
Date   : Jun 03 2017
Time   : 16:13:39
Host   : "gorina1.ux.uis.no"
PID    : 142670
Case   : /nfs/student/sveinen/height_change/West/W_6m
nProcs : 1
sigFpe : Enabling floating point exception trapping (FOAM_SIGFPE).
fileModificationChecking : Monitoring run-time modified files using timeStampMaster
allowSystemOperations : Allowing user-supplied system call operations

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
Create time

Overriding DebugSwitches according to controlDict
    crossFlowTurbineALSource 0;

    actuatorLineSource 0;

    actuatorLineElement 0;

    LeishmanBeddoes 0;

    LeishmanBeddoes3G 0;

Creating block mesh from
    "/nfs/student/sveinen/height_change/West/W_6m/system/blockMeshDict"
Creating curved edges
Creating topology blocks
Creating topology patches

Creating block mesh topology

Check topology

        Basic statistics
                Number of internal faces : 0
                Number of boundary faces : 6
                Number of defined boundary faces : 6
                Number of undefined boundary faces : 0
        Checking patch -> block consistency

Creating block offsets
Creating merge list .

Creating polyMesh from blockMesh
Creating patches
Creating cells
Creating points with scale 1
    Block 0 cell size :
        i : 1.25 .. 1.25
        j : 1.25 .. 1.25
        k : 1.25 .. 1.25


Writing polyMesh
----------------
Mesh Information
----------------
  boundingBox: (-50 -50 0) (50 50 20)
  nPoints: 111537
  nCells: 102400
  nFaces: 316160
  nInternalFaces: 298240
----------------
Patches
----------------
  patch 0 (start: 298240 size: 1280) name: walln
  patch 1 (start: 299520 size: 1280) name: walls
  patch 2 (start: 300800 size: 6400) name: ground
  patch 3 (start: 307200 size: 1280) name: inlet
  patch 4 (start: 308480 size: 1280) name: outlet
  patch 5 (start: 309760 size: 6400) name: top

End
```

9.4 Case files

9.4.1 fvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         backward;
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss linearUpwind grad(U);
    div(phi,k)      Gauss upwind;
    div(phi,epsilon) Gauss upwind;
    div(phi,R)      Gauss upwind;
    div(R)          Gauss linear;
    div(phi,nuTilda) Gauss upwind;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p;
}


// ************************************************************************* //
```

9.4.2 fvOptions

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvOptions;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


turbine
{
    type            crossFlowTurbineALSource;
    active          on;

    crossFlowTurbineALSourceCoeffs
    {
        fieldNames          (U);
        selectionMode       cellSet;
        cellSet             turbine;
        origin              (14.3 5.3 14);  // Center wind turbine
        axis                (0 0 1);        // Negative for opposite rotation
        rotorRadius         0.5;
        freeStreamVelocity  (0 -5.7 0);
        tipSpeedRatio       1.9;
        tsrAmplitude        0.0;            // Amplitude for TSR oscillation
        tsrPhase            1.4;            // Angle of first peak (rad)
        addedMass           on;

        dynamicStall
        {
            active              on;
            dynamicStallModel   LeishmanBeddoesSGC;
            LeishmanBeddoesSGCCoeffs
            {
                calcAlphaEquiv off;
                Tp          1.7;        // Default = 1.7
                Tf          3.0;        // Default = 3.0
                TAlpha      6.25;
                alphaDS0DiffDeg 3.8;
                r0          0.01;
                Tv          11;
                Tvl         8.7;
                B1          0.5;
                eta         0.98;
                E0          0.16;
            }
        }

        flowCurvature
        {
            active              on;
            flowCurvatureModel  Goude; // Goude || MandalBurton
        }

        blades
        {
            blade1
            {
                writePerf   true;
                nElements   7;
                endEffects  on;
                elementProfiles (NACA0020);
                elementData
                ( // axialDistance, radius, azimuth, chord, chordMount, pitch
                    (-0.5 0.5 0.0 0.14 0.5 0.0)
                    ( 0.5 0.5 0.0 0.14 0.5 0.0)
                );
            }
            blade2
            {
                $blade1;
                writePerf   false;
                azimuthalOffset 120.0;
            }
            blade3
            {
                $blade2;
                azimuthalOffset 240.0;
            }
        }

        struts
        {
            strut1
            {
                writePerf   true;
                nElements   4;
                elementProfiles (NACA0020 NACA0020 NACA0020 corner);
                elementData
                ( // axialDistance, radius, azimuth, chord, chordMount, pitch
                    (0.0 0.05 0.0 0.14 0.3 0.0)
                    (0.0 0.5 0.0 0.14 0.3 0.0)
                );
            }
            strut2
            {
                $strut1;
                writePerf   false;
                azimuthalOffset 120.0;
            }
            strut3
            {
                $strut2;
                azimuthalOffset 240.0;
            }
        }

        shaft
        {
            nElements   7;
            elementProfiles (cylinder);
            elementData
            ( // axialDistance, diameter
                (-0.66 0.09)
                ( 0.66 0.09)
            );
        }

        profileData
        {
            NACA0020
            {
                tableType   multiRe; // singleRe || multiRe
                Re          1.6e5;
                data        (#include "NACA0021_1.6e5");
                ReList      (4e4 8e4 1.6e5 3.6e5 7e5 1e6 2e6 5e6);
                clData      (#include "NACA0021_multiRe_cl");
                cdData      (#include "NACA0021_multiRe_cd");
                cmData      (#include "NACA0021_multiRe_cm");
            }
            cylinder
            {
                data
                (
                    (-180 0 1.1)
                    (180 0 1.1)
                );
            }
            corner
            {
                data
                (
                    (-180 0 0.05)
                    (180 0 0.05)
                );
            }
        }
    }
}

// ************************************************************************* //
```